Content-Based Retrieval of Melodies using Artificial Neural Networks

Steven Harford, B.Sc.

A dissertation submitted for the degree of Doctor of Philosophy under the supervision of Dr Mark Humphrys at the School of Computing, Dublin City University.

September 2006

This dissertation is dedicated to my wife, Fiona. Thanks for believing in me!

In loving memory of my grandmothers, Edith Mansfield and Nancy Harford.

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Steven Harford

ID No.: 93382634

Date: 20 September 2006

Acknowledgements

Producing this dissertation has been a journey, one which encompassed a myriad of seemingly impossible impasses and their eventual resolution. I could never have begun, let alone completed, such a journey on my own.

In acknowledgement, I would like to thank Dublin City University and the School of Computing for their persistent support of this research. In particular, I would like thank my supervisor, Dr Mark Humphrys, for helping me navigate my way through this work, for many insightful comments and for a few, much-needed, pep talks! I am also grateful to Dr Mike Page and Prof. Alan Smeaton for taking the time to read this thesis and for their comments and suggestions. In addition, I would like to acknowledge the Music Information Retrieval and Artificial Neural Network communities for their remarkable and ever-increasing body of work, upon which this research was built and inspired.

I consider it a tremendous privilege to have been in a position to pursue this postgraduate degree. For this opportunity I am eternally grateful to my parents. I could not have wished for more. Finally, to my wife Fiona, the driving force behind this effort. Not only has she patiently reviewed every word of this dissertation, she has tirelessly and selflessly supported and stood by me throughout — thank you!

Contents

Abstract

Human listeners are capable of spontaneously organizing and remembering a continuous stream of musical notes. A listener automatically segments a melody into phrases, from which an entire melody may be learnt and later recognized. This ability makes human listeners ideal for the task of retrieving melodies by content. This research introduces two neural networks, known as SONNET-MAP and *Re*TREE*ve*, which attempt to model this behaviour. SONNET-MAP functions as a melody segmenter, whereas *Re*TREE*ve* is specialized towards content-based retrieval (CBR).

Typically, CBR systems represent melodies as strings of symbols drawn from a finite alphabet, thereby reducing the retrieval process to the task of approximate string matching. SONNET-MAP and *Re*TREE*ve*, which are derived from Nigrin's SONNET architecture, offer a novel approach to these traditional systems, and indeed CBR in general. Based on melodic grouping cues, SONNET-MAP segments a melody into phrases. Parallel SONNET modules form independent, sub-symbolic representations of the pitch and rhythm dimensions of each phrase. These representations are then bound using associative maps, forming a two-dimensional representation of each phrase. This organizational scheme enables SONNET-MAP to segment melodies into phrases using both the pitch and rhythm features of each melody. The boundary points formed by these melodic phrase segments are then utilized to populate the *Re*TREE*ve* network.

*Re*TREE*ve* is organized in the same parallel fashion as SONNET-MAP. However, in addition, melodic phrases are aggregated by an additional layer; thus forming a two-dimensional, hierarchical memory structure of each entire melody. Melody retrieval is accomplished by matching input queries, whether perfect (for example, a fragment from the original melody) or imperfect (for example, a fragment derived from humming), against learned phrases and phrase sequence templates. Using a sample of fifty melodies composed by *The Beatles*, results show that the use of both pitch *and* rhythm during the retrieval process significantly improves retrieval results over networks that only use either pitch *or* rhythm. Additionally, queries that are aligned along phrase boundaries are retrieved using significantly fewer notes than those that are not, thus indicating the importance of a human-based approach to melody segmentation. Moreover, depending on query degradation, different melodic features prove more adept at retrieval than others.

The experiments presented in this thesis represent the largest empirical test of SONNET-based networks ever performed. As far as we are aware, the combined SONNET-MAP and *Re*TREE*ve* networks constitute the first self-organizing CBR system capable of automatic segmentation and retrieval of melodies using various features of pitch and rhythm.

Part I

General Background

Chapter 1

Introduction

This chapter provides a general overview of this thesis, its contributions and a guide to how to read this work.

1.1 Content-based retrieval of melodies

In general terms, *content-based retrieval* (CBR) of melodies involves measuring the similarity of an input query, which consists of a melody fragment, against a database of melodies. The aim is to produce a ranked list containing the best matches. The ability to recognize and retrieve melodies in this way has a variety of uses. For example, a consumer may wish to locate a particular piece of music for which he or she has no textual information (for example, the melody's title or its composer). However, if the consumer can reproduce a fragment from the piece in question (for example, by reproducing the chorus or a verse, which may be achieved by humming, playing a musical instrument or by composing the query directly using tailored computer software), the use of a CBR system may help the consumer track down the music they have been looking for. This is particularly useful for record companies trying to boost sales, whether on the high street or over the internet. For music rights organizations, this technology can aid in protecting artists from plagiarism and in detecting unlawful broadcasts. Musicologists are also interested in studying melodic similarity, perhaps in establishing the resemblance of *Oasis* compositions to those of *The Beatles.*

To date, most CBR research has approached this problem by representing a melody (both the query and the database entries) as a string of symbols that are drawn from a finite alphabet. This effectively reduces the retrieval process to the task of approximate string matching. Such approaches fail to acknowledge the inherent hierarchical structure of melodies and the wealth of melodic information available to infer such a structure. This dissertation introduces a novel approach to typical, and indeed all, CBR research, challenging many common assumptions made by such systems; in particular the creation, representation, organization and retrieval of melodic information.

1.2 Human abilities in melody retrieval

Human listeners are capable of spontaneously organizing and remembering a continuous stream of musical notes that comprise a melody. When recalling a melody, a listener cannot usually begin at an arbitrary point in the melody. Instead, a memorable place such as the beginning of a verse or a chorus becomes the starting point. Research in music psychology strongly suggests that these memorable points are determined using perceptual grouping cues. The term *boundary point* is used to describe these memorable places in a melody and the process of determining boundary points is called *melody segmentation*. The *segments* (or *phrases*) that form between consecutive boundary points are further organized in a hierarchical fashion, which facilitates the learning of entire melodies (we review the evidence for this in Chapter 2). Furthermore, familiar melodies can be robustly recognized at a later time, even if these melodies are not accurate reproductions of the originals. These extraordinary abilities make human listeners ideal for the task of CBR.

Research into these abilities suggest possible algorithmic and structural approaches to CBR research. This dissertation introduces a *human-inspired* model that acknowledges and applies these suggestions. Although the design and function of this model is based on aspects of music psychology and neuroscience, it is *not* claimed that this model implements any of the underlying neural mechanisms of a human listener's brain. In any case, whether or not the studies that this human-inspired approach is based upon are believed to be plausible, the empirical results presented in Chapters 12 and 13 demonstrate that this model stands on its own as an engineering solution, producing useful results and insights into cognitive systems and CBR research.

1.3 Symbolic Artificial Intelligence for music processing

With the invention of digital computers during the twentieth century, the study of human cognition in the form of Artificial Intelligence (AI) was born. Early AI research, fuelled by the von Neumann computer architecture, was dominated by the tenet that the mind is a symbol system whereby human thought involves the manipulation of symbols prescribed by formal rules. This hypothesis, known as the *symbol system hypothesis*, forms the basis of symbolic AI and has had successes ranging from household chess software employing AI search techniques [9, 86, 93] to expert systems capable of diagnosing medical illnesses [30, 116, 132, 150].

This formal, rule-based methodology has also been employed by early models of music cognition. Although symbolic-based approaches have achieved some success in this area, there are growing concerns among music theorists regarding certain drawbacks that are inherent in this approach.

> *"It seems that traditional techniques for modeling intelligence do not fare well when applied to deep humanistic problems such as music."* [90, pg. 29]

Typical rule-based music research (and indeed symbolic AI) has been necessarily restricted to simple, predictable and relatively static toy musical environments. Systems developed using rule-based technology have difficulty generalizing knowledge from their input. Consequently, domain specific heuristics are hard-coded to aid in problem solving. The manner by which these heuristics, and indeed knowledge in general, can be autonomously learned and retrieved presents other significant drawbacks. In other words, rule-based symbol systems have difficulty in adequately addressing the *knowledge problem* [30, pgs 91–92]. Furthermore, these systems tend to have trouble with noisy input signals, which are common in music. For example, comparing a concert rendition of a song (or indeed a hummed fragment) against its recorded counterpart. These shortcomings are significant if we intend to model the real-time, autonomous, adaptable and robust behaviour of a human listener. An alternative approach, in the form of Artificial Neural Networks (ANNs), offers novel insights in addressing these issues.

1.4 Artificial Neural Networks for music processing

During the mid to late 1980's, with the rebirth of ANNs[1], mainly due to Rumelhart and McClelland [137, 138], research into music cognition took on a fresh approach. ANNs seem to tackle, head on, the stumbling blocks encountered by the traditional rule-based techniques. In particular, ANNs were reported that could passively acquire knowledge through principles of self-organization with little or no domain specific knowledge. Furthermore, these networks have proven especially adept at processing noisy input patterns. Although ANN technology is still in its infancy, the intrinsic advantages of this approach are compelling. As a consequence, many music theorists have now embraced ANNs, and the results of many new studies have been reported. Areas of research into music cognition using ANNs include pitch and rhythm perception, quantization of musical time, tonal analysis, melodic memory and music composition [60, 160] (for more detail see Chapter 4). The use of ANNs, motivated by the analytical top-down and bottom-up disciplines of music psychology and neuroscience respectively, will shape the human-inspired approach to CBR proposed herein.

1.5 SONNET and melody processing

Carpenter and Grossberg [14, 15, 16] introduced a family of ANNs, referred to under the umbrella of Adaptive Resonance Theory (ART), to address the problem of how to build stable, real-time, autonomous, adaptable and robust systems. A known problem with these networks is that they are unable to recognize patterns that are embedded within larger patterns or contexts. Therefore patterns must be segmented before they are offered up to these networks [114, 118]. Since in real-world environments patterns are unsegmented, this poses serious questions regarding the ability of these networks to behave autonomously. Cohen and Grossberg [28] introduced a type of network, known as a *masking field*, to address the problem of recognizing embedded patterns. However,

[1] Also known as connectionism or parallel distributed processing. However, the term *sub-symbolic AI* can be used to cover all of these terms.

masking fields are unable to form stable category codes[2], which is a fundamental property of any intelligent agent [114, 118]. Nigrin [112] introduced a network called SONNET[3] (**S**elf-**O**rganizing **N**eural **NET**work) that combines the stability of ART and the capacity of masking fields to recognize embedded patterns. Although Nigrin was inspired to invent SONNET for processing letter sequences, almost all SONNET research to date has been applied to musical sequence processing [88, 118, 133].

Melodic input can be extracted from a raw acoustic source (such as humming) or from various file formats (for example: WAV, MP3, WMA and MIDI). The extraction of melodic information from these sources is beyond the scope of this thesis. Therefore, preliminary analysis of a melodic source is assumed to already have been performed by a subsystem capable of extracting pitch and timing information. In other words, in this thesis, SONNET processes sequences of abstract musical events that are represented in much the same way as in a musical score. That is, each note event is represented by a duple containing its pitch and duration. Indeed, most neural networks that perform high-level tasks, such as learning, recall, composition or tonal analysis, work with a similarly high-level melody representation, and make the same assumption (including previous SONNET-based research and, indeed, much CBR research). For more information on melody representations, refer to Chapters 2, 3 and 9.

So, melodies are presented to SONNET as sequences of pitch events that occur at particular points in time. SONNET's short-term memory (STM) stores these event occurrences and the order in which they occur. Then, based on melodic grouping cues and principles of self-organization, SONNET can segment a melody into significant phrases, which are learnt by adaptable connection weights. Therefore, long-term memory (LTM) is populated with melodic phrases that are represented sub-symbolically by the strength of the network's connections. This approach is markedly different from most CBR systems, which simply represent melodies as strings of symbols. For example, a *melodic contour* representing five pitches may be specified using the following sequence of symbols — UUSD, where U represents an increasing interval, D represents a decreasing interval and S represents a repeated pitch. Although some systems do attempt to segment melodies, such as the *n-gram* style of indexing used for searching, very rarely are a melody's features used to aid in this segmentation process or, consequently, in the search process.

During preliminary experimentation with SONNET using melodic input, a number of limitations and problems with the incumbent formulation were identified (see Chapter 8). Firstly, SONNET cannot process phrase repetitions adequately. Consequently, a mechanism which enables presynaptic inter-cell competition to handle phrase repetitions properly was introduced. Secondly,

[2] The pattern learned by a network at a particular node is known as a category code. Category codes are also referred to as categorical representations, compressed representations, unitized representations or chunks.

[3] Nigrin's original network may be referred to as either SONNET or SONNET 1 [110, 111, 112, 114]. Although Nigrin also introduced SONNET 2 (but did not actually implement it [114, pg. 317]), all subsequent SONNET-based research (including the work presented in this thesis) has been primarily based on Nigrin's original network. In this dissertation, the use of SONNET and SONNET 1 are considered interchangeable.

a new multiplicative confidence formulation was introduced which, unlike previous multiplicative formulations, adequately measures arbitrary levels of mismatch between STM and LTM.

1.6 SONNET-MAP and melody segmentation

Deutsch [36, pg. 366] points out that *"Music theorists have argued that the tonal music of our tradition is composed of segments that are organized in hierarchical fashion"*. Page [118, 119] shows how SONNET modules can be cascaded to form a melodic phrase hierarchy, where phrases consist of *scale notes*. Higher level modules aggregate phrases from lower level modules in order to construct such a hierarchy. This configuration aids in melody recognition and, in particular, melody recall. Recognition and recall can be further improved if both pitch and rhythm are employed. Lewis [88] brings both of these dimensions to bear on the problem of interpreting melodic structures by combining scale notes with an *inter-onset interval* (IOI) representation using a single SONNET network. However, this approach does not allow the recognition of a rhythm independently of its corresponding pitch sequence and vice versa. This is a serious drawback and is clearly at odds with human melody recognition. Moreover, this approach does not provide a framework for the addition of other melodic features such as *pitch-intervals* or melodic contours.

Indeed, a recent study of perceptual musical functions in patients suffering from unilateral cerebrovascular cortical lesions indicated:

> *"a perceptual independence of melody and rhythm, and [that] it seems reasonable to assume that they are processed by separate neural subsystems which may be closely linked or partially integrated."* [143, pg. 557]

Interestingly, such findings support the *K-line* theory of memory put forward in Minsky's famous *Society of Mind* thesis [104], where a K-line attaches itself to the different functional agents that were active at the time something memorable was experienced.

Page [118, pg. 317] postulates the existence of parallel hierarchies, suggesting that *"the interaction between parallel hierarchies, describing different aspects (pitches, rhythm etc.) of the same stimulus sequence, needs to be investigated."*. Indeed, Roberts [133, pg. 120] draws attention to the fact that *"more information can be extracted from the input if different input dimensions are processed individually"*. Nevertheless, the manner by which such an organization can be achieved has hitherto undergone little investigation.

This dissertation introduces a network, known as SONNET-MAP, which addresses the issues raised above. SONNET-MAP is composed of two parallel SONNET modules; one which processes pitch sequences and another which processes rhythms. These modules are connected via associative maps, derived from the ARTMAP architecture [21], which bind pitch sequence representations with their corresponding rhythm representations. These associations allow the construction of rigid, two-dimensional representations of melodic phrases. In addition, a further SONNET module may be utilized to aggregate these phrases, forming a parallel and hierarchical memory structure that encompasses the entire melody. Unlike Lewis' approach, pitch sequences and rhythms can be recognized independently of one another while at the same time, due to the associations learned between pitch and rhythmic phrases, both dimensions can be utilized during the segmentation process.

In this thesis, SONNET-MAP is utilized solely as a melody segmenter. Based on melodic grouping cues, SONNET-MAP segments a melody into phrases. Parallel SONNET modules form independent representations of the pitch and rhythm dimensions of each phrase which, as we have already mentioned, are then bound using associative maps. This organizational scheme enables SONNET-MAP to segment melodies into phrases using both the pitch and rhythm features of each melody. The boundary points formed by these melodic phrase segments are then utilized to populate the *Re*TREE*ve* network.

1.7 *Re*TREE*ve* and the CBR of melodies

During preliminary CBR testing, two weaknesses of using SONNET-MAP directly as a retrieval system were discovered (see Section 11.2.1). Firstly, retrieval run times were excessive using SONNET-MAP (see Section 11.2.2). This was primarily due to the requirement of using differential equations to model the passage of time, and to the implementation of this inherently parallel architecture on a traditional von Neumann sequential processor. Consequently, the segmentation process and the retrieval process were separated into distinct subsystems. SONNET-MAP was utilized to segment melodies (as described above), whereas *Re*TREE*ve* was developed as a specialized retrieval system. This logical and physical partitioning of function dramatically improved the performance of melody segmentation and retrieval.

Secondly, it was discovered that the *winner-take-all* competitive strategy, utilized in almost all competitive ANNs (including SONNET), fails to behave in a manner that allows the satisfaction of global constraints (see Section 11.2.3). In other words, the winner at one particular point in time does not necessarily emerge as the winner once the global context emerges. Therefore, the use of a *multiple winners* recognition strategy was employed. The use of multiple winners also facilitated simplifications of the retrieval subsystem's operational dynamics, which dramatically improved retrieval performance. More information on these and other issues can be found in Chapter 11.

*Re*TREE*ve* is organized in the same parallel fashion as SONNET-MAP. However, in addition, melodic phrases are aggregated by an additional layer, thus forming a two-dimensional, hierarchical memory structure of each entire melody. Unlike some CBR systems, *Re*TREE*ve* does not retrieve melodies using traditional string matching algorithms. Instead, the retrieval process is accomplished by matching input queries against previously learned phrases and phrase sequence templates. This is achieved using a combination of a multiple winners strategy and a *match-based* retrieval algorithm. A match-based retrieval algorithm, in the context of ANNs, involves measuring the similarity between melodic phrases, stored sub-symbolically as LTM connection weights, against the current melodic query, which is represented by input cell activation levels. The parallel and hierarchical structure of *Re*TREE*ve* provides a novel organizational approach to the CBR of melodies and offers several practical benefits (see Section 14.2.4). *Re*TREE*ve* also utilizes a match parameter, which is used to allow either exact or approximate melody matching.

1.8 Empirical results

The empirical tests presented in Chapters 12 and 13 employ a set of fifty melodies composed by *The Beatles*, and represent the largest test of SONNET-based neural networks ever performed. It was decided to use a set of melodies composed by this single group because we believe that the phrases formed from this set would be quite similar to one another. Consequently, given any individual query, it would be more difficult to discriminate between different melodies during retrieval.

The segmentations performed by SONNET-MAP are consistent with many aspects of melody perception. For example, most of the segments formed are based on grouping mechanisms such as proximity, repetition, the gap principle and the limitations of short-term and long-term memory. Consequently, in almost every case, the beginning of each verse or chorus is aligned along a phrase boundary, which is very desirable since these are the most memorable places of a melody, and the most likely places for a user to begin a query.

Utilizing the phrase segments formed using SONNET-MAP, *Re*TREE*ve*'s LTM is populated with the entire set of melodies. Then, a selection of melodic queries are presented to evaluate the *Re*TREE*ve* algorithm's performance. The retrieved melody associated with each query is compared against the expected result, at which point the accuracy of the retrieval can be measured. Queries are generated directly from the original melodies, with various transformations being performed on each query before it is presented for retrieval. These transformations modify a query in a manner that models the types of errors and imperfections that humans may make in formulating a query.

Results show that the use of both pitch *and* rhythm during the retrieval process significantly improves retrieval results when compared with networks that use only pitch *or* rhythm. This illustrates the necessity of using multidimensional hierarchies over standard, single-dimensional hierarchies and, more generally, that as much information as is available should be utilized by a CBR system. Furthermore, queries that coincide with learned boundary points are retrieved using significantly fewer notes than those that are not. This indicates the importance of a human-based approach to melody segmentation. Moreover, depending on the extent of query degradation, different melodic features prove more adept at retrieval than others.

1.9 Summary of original contributions of this thesis

This section briefly describes some of the major contributions of this thesis.

STM and LTM representation of melodies. A shared event field at $\mathcal{F}_0$ is introduced, which temporarily records the notes of a melody as they are presented. $\mathcal{F}_0$ contains 88 cells, each responding to a particular *categorical pitch-height* (CPH), spanning low A to high C. Therefore, $\mathcal{F}_0$ acts like a tonotopically organized echoic memory. Methods are presented that show how certain characteristic features of a melody — IOIs, invariant pitch-classes (IPCs), pitch-intervals and melodic contours — can be derived from this event field and represented at separate SONNET short-term memories. To help achieve this, generalized input field dynamics are introduced. In particular, a new method of representing pitch-intervals and melodic contours using these dynamics is presented. Finally, the manner by which SONNET stores melodies in LTM is illustrated. Although the storing of melodies using connection strengths is not new with respect to ANNs, it is a novel approach to the CBR of melodies.

Improvements to SONNET. A method of using presynaptic inhibition for adequately handling phrase repetitions (particularly overlapping phrase repetitions) is introduced. Furthermore, a new multiplicative SONNET confidence formulation is presented which can measure arbitrary levels of mismatch between STM and LTM.

Introduction of SONNET-MAP. We introduce the SONNET-MAP neural network, which has a parallel (and hierarchical) configuration. This allows different melodic features to be processed separately and simultaneously. Furthermore, a number of synchronization techniques that facilitate this parallel organization are introduced. These techniques involve the learning of associations between pitch sequences and their corresponding rhythms. These associations can be utilized to perform associative priming, which significantly aids in the segmentation process.

Introduction of *Re*TREE*ve*. We introduce the *Re*TREE*ve* network, which is derived from SONNET-MAP and can be used as an efficient, robust and accurate retrieval system for melodies. *Re*TREE*ve*'s organizational structure and operation provide a novel approach to melody retrieval.

Empirical findings — SONNET-MAP. We show that SONNET-MAP is capable of discovering plausible boundary points within a melody using a pitch-interval and IOI representation, where grouping cues such as proximity, repetition, the gap principle and the limitations of short-term and long-term memory aid in the segmentation process.

Empirical findings — *Re*TREE*ve*. Using *Re*TREE*ve*, it is shown that (a) the combination of two dimensions (pitch and rhythm) significantly improves retrieval performance over the use of a single dimension (either pitch or rhythm), (b) queries that are aligned along phrase boundaries are retrieved more effectively than queries that are not aligned along phrase boundaries, thus providing evidence suggesting the importance of utilizing human-based segmentation techniques, (c) particular melody representations are more suited to particular query types, and finally, (d) retrieval performance degrades gracefully as the quality of the queries degrade.

As far as we are aware, SONNET-MAP and *Re*TREE*ve* constitute the first self-organizing neural network system capable of automatically segmenting and retrieving melodies based on the combination of pitch and rhythm [65].

1.10 A brief guide to this thesis

Chapters 2–4 provide background information on music psychology, music information retrieval (MIR) and the application of ANNs to computer music. Readers familiar with all of these topics may wish to go directly to Chapter 5.

Chapters 5–7 provide the necessary background on the fundamentals of SONNET, advanced SONNET topics and recent applications of SONNET to melody processing. Once again, readers familiar with this research may wish to proceed directly to Chapter 8.

Chapter 8 introduces a number of improvements that we have made to SONNET, which generally improve the network's operation and are vital to the work presented in later chapters on configuring SONNET in a parallel fashion.

Chapter 9 describes how melodies can be stored in SONNET's STM as cell activations using generalized input field dynamics. This chapter argues that each dimension of a melody should be processed by an independent, parallel, working memory. This argument is a precursor to the work presented in Chapter 10. We also illustrate how melodies are encoded in SONNET's LTM by the strength of connection weights.

Chapter 10 introduces the SONNET-MAP neural network and the synchronization techniques required to facilitate its parallel (and hierarchical) organization.

Chapter 11 introduces the *Re*TREE*ve* network and shows how it can be utilized as a CBR system for melodies. In particular, efficiency considerations and the inadequacies of winner-take-all competitive fields for retrieval are discussed and addressed.

Chapter 12 presents experimental results that illustrate how SONNET-MAP can segment melodies using grouping cues from both the pitch and rhythm dimensions of a melody. The boundary points formed during this segmentation process are used to populate *Re*TREE*ve*.

Chapter 13 presents experimental results which illustrates *Re*TREE*ve*'s performance at retrieving melodies by content.

Chapter 14 summarizes the contributions of this research and suggests avenues of further investigation for SONNET-based research, melody segmentation and retrieval, applications in music processing and other disciplines.

Appendices A–F provide specifications of the SONNET, SONNET-MAP and *Re*TREE*ve* networks. An overview of some of the issues encountered during their implementation is also provided. Furthermore, the catalogue of melodies used during the empirical testing is presented. Finally, a glossary is provided that summarizes the principal parameters used in this thesis.

Chapter 2

Melody: Perception and Memory

2.1 Introduction

Intelligent behaviour can be characterized by its autonomy, adaptability, robustness and real-time operation within unconstrained, unpredictable, complex and dynamic environments. This dissertation argues the case for intelligent melody retrieval, which is inspired by research that suggests how humans achieve this behaviour. In particular, humans can spontaneously create and organize melodic memories, and subsequently recognize these melodies reliably. Therefore, to lay a necessary foundation for a human-inspired approach, this chapter discusses a variety of aspects relating to the human perception and memory of melodies.

A melody consists of a sequence of musical tones (or sounds) that occur sequentially at particular points in time. In contrast to noises, which consist of irregular and disordered sound wave vibrations, tones consist of regular and uniform vibrations, or periodic sound waves. These mechanical sound waves are converted into electrical nerve impulses by our ears and are further processed by the remaining components of our auditory system (see Section 2.2). A tone may be described in terms of the perceptual characteristics of pitch, intensity (loudness) and quality (timbre), which correspond directly to the following physical characteristics of a sound wave: frequency, amplitude and harmonic constitution (or waveform). The most characteristic feature of a tone is its pitch, which will be discussed in greater length in Section 2.3. Since tones occur at particular points of time, there exists a temporal dimension to melody. This temporal dimension, known as rhythm, will be discussed in Section 2.4. Additionally, the perceptual and cognitive organization of melodies is also discussed; in particular memory and grouping mechanisms (see Sections 2.5, 2.6, 2.7 respectively). Since each of these topics envelop an enormous body of work, only aspects which are pertinent to this thesis will be discussed.

2.2 Hearing

A sound is produced when an object vibrates, causing a disturbance which travels through an elastic medium. Sound can travel through mediums such as earth, water or air (but not a vacuum).

Typically, the sounds heard by the human ear are consequences of objects that vibrate in air. For example, a musician plucking a guitar string. The vibration of an object in air results in a series of alternate compressions and rarefactions of the surrounding air molecules, which produce a sound wave. Since the energy of sound waves emanate out from the centre of the source vibration, sound waves are longitudinal.

The ear, which is the organ of hearing (and balance), is constructed of three distinct components: the air-filled outer and middle ears, and the fluid-filled inner ear. The ear converts the sound waves produced by vibrating objects into electrical impulses, which are sent via the auditory pathways to the auditory cortex where they are interpreted. Greater detail on the material presented in this section may be found in [74, Chapter 6], [115, Chapter 14] and [94, 95, 171].

2.2.1 The outer ear

The outer ear is primarily comprised of the pinna (or auricle) and the external auditory canal (or meatus). Its function is to catch and deflect sound waves to the middle ear. The outer ear also helps in determining the directional location of a sound source.

2.2.2 The middle ear

The middle ear is primarily comprised of the tympanic membrane (or eardrum) and a chain of three small moveable bones called the ossicles. The ossicles consist of the malleus (or hammer), incus (or anvil) and stapes (or stirrup). These are the smallest bones in the human body.

The compressions and rarefactions of sound waves, which are conducted by the external auditory canal, cause the eardrum to vibrate back and forth (analogous to the operation of a microphone's diaphragm). The eardrum is rigid and extremely sensitive. For instance, at approximately 3000 Hz (the frequency that humans are most sensitive to) the eardrum moves a distance of less than the diameter of a single hydrogen atom (Nolte [115, pg. 331]). These sound induced vibrations are transferred along the ossicles to the inner ear, which ultimately results in movement of the fluid in the inner ear. Since the inner ear conducts sound through a fluid, and since fluid is harder to move than air, the vibrations must be amplified, which is the function provided by the middle ear.

2.2.3 The inner ear

Primarily, the inner ear consists of the semicircular canals and the cochlea. The semicircular canals are responsible for our sense of balance, while the cochlea is responsible for our sense of hearing. The cochlea (Greek for snail shell) has a snail-like structure and is filled with a fluid called endolymph. The cochlea consists of three adjacent tubes: the scala vestibuli, the scala media and the scala tympani. The basilar membrane separates the scala vestibuli and the scala media from the scala tympani. The basilar membrane consists of approximately 20,000 to 30,000 reed-like fibers extending the width and length of the cochlea. Fibers are short (approximately 100 μm wide) and taut near the oval window (at the base of the cochlea) and gradually get longer (approximately 500 μm wide) and more supple towards the round window (at the apex of the cochlea). Consequently, different areas of the basilar membrane have different resonant frequencies. Resting on the basilar membrane is the organ of Corti which is approximately 35mm long and contains approximately 30,000 specialized neurons or receptors called hair cells.

As we have already mentioned, sound waves are transferred from the air-filled outer and middle ears to the fluid-filled inner ear. This transference occurs at the junction where the stapes, which occupies a hole in the temporal bone called the oval window, meets the scala vestibuli. The back and forth movement of the stapes against the oval window membrane invokes pressure waves in the cochlea, similar to ripples on a pond. These waves are propagated along the basilar membrane, growing in amplitude to a peak and then collapsing at the point where the fibres of the basilar membrane have the same resonant frequency as the sound wave. As a consequence, the hair cells of the organ of Corti within the receptive field of the collapsed wave will move, causing them to activate and send electrical impulses though the cochlea nerve. These impulses are sent, via the auditory pathways, to the auditory cortex where they are interpreted by the brain to determine perceptual characteristics such as pitch and intensity. The tuning characteristics of the basilar membrane represent the beginning of a tonotopic organization mirrored by the relay nuclei of the auditory pathways and the auditory cortex. It is important to note that similar mappings exist for other perceptual systems. For example, the somatotopic organization of the somatosensory systems. This suggests that some underlying principle governing brain plasticity is involved in the mapping of sensory information from the its raw stimulus to its corresponding areas of the cerebral cortex. Indeed, much ANN research has studied principles of brain plasticity in the form of self-organizing feature maps [72].

2.2.4 Auditory pathways

Generally, the axons of hair cells exit the cochlea and form the major part of the auditory nerve (or VIIIth cranial nerve). Auditory information is conveyed by the auditory nerve to the auditory cortex via several nuclei and pathways at various levels in the brain stem. These include the cochlear nucleus, the superior olivary nucleus, the trapezoid body, the lateral lemniscus, the inferior colliculus, the inferior brachian, and the medial geniculate nucleus. Each of these areas consists of several constituent nuclei. In particular, the medial geniculate nucleus consists of three major subdivisions: the ventral, the dorsal and the medial/magnocellular. Furthermore, the tonotopic organization of the cochlea is reflected throughout the the auditory system, with both ascending and descending pathways contributing to processing auditory stimuli.

2.2.5 Auditory cortex

In general, the auditory cortex consists of several primary cortical fields (A1 or Brodman's area 41) located in the transverse temporal gyri (of Heschl) and is surrounded by several secondary areas. The subdivision of the medial geniculate nucleus projects to several different areas of the auditory cortex, which is consistent with the *"general pattern of multiple independent ascending pathways to* *the cortex*" (Kolb and Whishaw [74, pg. 111]). The ventral subdivision is tonotopically organized and projects to several fields in the primary auditory cortex which are similarly tonotopically organized. The dorsal subdivision projects to the secondary regions of the auditory cortex, which are not thought to be tonotopically organized (nontonotopic). Finally, the medial/magnocellular subdivision, although nontonotopic, projects to all areas of the auditory cortex.

Although this dissertation does not claim in any way that the systems introduced (SONNET-MAP and *Re*TREE*ve*) are accurate models of any aspects of the auditory system, it does however claim to utilize various structural ideas. For example, Chapter 9 introduces a tonotopically organized event field ($\mathcal{F}_0$). Additionally, the various levels that exist from the auditory nerve to the auditory cortex suggests structuring the architecture of any human inspired systems in a layered fashion; this approach is also utilized by SONNET-MAP and *Re*TREE*ve* ($\mathcal{F}_0$, the various gating mechanisms, the input fields and the masking fields). Moreover, the separation of the input stimulus into parallel and independently organized cortical fields is a major tenet of the SONNET-MAP and *Re*TREE*ve* networks (see Chapters 10 and 11).

2.3 Pitch

2.3.1 Perceived pitch

The frequency of a sound wave is defined by the number of cycles (or oscillations) it completes in a given length of time, which is measured in hertz (cycles per second). The more rapidly a sound wave vibrates, the higher the perceived pitch is. A tone may be simple in that its sound wave is just sinusoidal [126]. The perceived pitch of such tones directly correlates to its frequency. However, more generally, tones are complex. A complex tone consists of several frequency components, a *fundamental frequency* labelled f_0, and a set of *partials* or *harmonics* labelled $2f_0$, $3f_0$, $4f_0$, $5f_0$ and so forth. All of the frequency components of a complex tone are fused into the sensation of a single pitch percept (provided the acoustic vibrations occur fast enough), in which the perceived pitch is that of the fundamental frequency. This is true even if the fundamental frequency is absent from the tone [125, 127]. In this case, the fundamental frequency is known as the *virtual pitch* or *residue pitch*. Therefore, since most tones we hear are complex, a pitch perception process exists that not only fuses the partials of a complex tone into the sensation of a single pitch percept, but additionally identifies the perceived pitch as that of the fundamental frequency regardless of its presence or absence.

The manner by which this process is achieved has given rise to a variety of pitch perception theories. In particular, cochlea-based theories such as Helmholtz's place theory [164] and Schouten's periodicity pitch theory [142]. However, later experimental investigations have shown that the cochlea alone does not account for our ability to determine the pitch of a musical tone [69]. Thus, in addition to the cochlea, other areas of the auditory system must be involved in pitch determination. Consequently, more recent research has focussed on pattern recognition theories.

In particular, De Boer's template matching scheme [35] and its various elaborations, including: Goldstein's optimum processor theory [58], Terhardt's virtual-pitch theory [156] and Wightman's pattern-transformation theory [173]. Although a variety of implementations of these theories have been published, according to Taylor and Greenhough [155, pg. 6] *"these implementations generally involve complex algorithms whose success is often qualified by the need to set parameters in an ad hoc fashion, so that the results fit empirical data from psychoacoustic experiments"*. Consequently, recent research has focussed on the development of ANN models of pitch perception (see Chapter 4).

2.3.2 Limits and range of pitch perception

Humans can hear sound vibrations between 20Hz and 20,000Hz, although the upper limit reduces with age to 16,000Hz at middle age to as low as 8,000Hz for a person aged 85. However, for an accurate sense of pitch (called the range of usable pitches [151, pg. 126]) the upper limit of this range is approximately between 4,000Hz and 5,000Hz. Furthermore, within this range lies the dominance region for pitch perception [131, pg. 96] (also called the *range of best pitch discrimination* [151, pg. 125]), which is approximately between 500Hz to 2,000Hz. It is within this range that humans can discriminate between different pitches most accurately.

Within the range of best pitch discrimination, humans can perceive over 1,400 distinct pitches. In contrast, however, Western music only utilizes approximately 88 discrete pitches, which is typical of most tuning systems. This difference can be explained in terms of two different tasks: a pitch discrimination task and a pitch identification task. In a pitch discrimination task, two different pitches are sounded in immediate succession and the listener is asked to decide whether these pitches are identical or not. Differences of approximately 0.3% between adjacent pitches can be discriminated. This difference is referred to as the *just noticeable difference* (JND). In a pitch identification task, listeners are again asked to state whether two pitches are different or not. However, in this task, intervening pitches exist between the pitches to be judged. At this task, significantly fewer pitches can be discriminated. In the pitch discrimination task absolute pitches are directly compared in echoic memory. However, for the pitch identification task, since intervening pitches exist, STM must be utilized because direct comparison is impossible. Consequently, rather than comparing directly the lingering pitch stimuli in echoic memory, categorical pitch representations must be compared, of which there are significantly fewer. Therefore small pitch variations in absolute pitch are discarded. (see Section 9.2.1 with respect to how this information is utilized in determining the architecture of $\mathcal{F}_0$).

Outside of these ranges, we do not have a particularly good sense of pitch. Below 20Hz we perceive acoustic vibrations as a buzzing noise. As the number of cycles per second increases, a transition range exists, known as the threshold of event fusion, where individual acoustic vibrations are not heard individually but are fused into a single pitch. Therefore, not only are the individual harmonic components of a pitch fused, so are the regular vibrations which constitute a pitch. As we move to the upper limits of hearing, particularly tones above 5,000Hz, tones are increasingly heard as brighter, rather than higher, in pitch. Furthermore, a tone containing only a few cycles has no perceived sense of pitch, but instead sounds like a click. For low frequencies the click sounds dull, whereas for high frequencies, the click sounds bright.

In summary, a pitched event is a grouping of acoustic vibrations which occur quicker than the threshold of event fusion. Changes in pitch indicate boundaries between two differently pitched events. This form of grouping is based on proximity and similarity which, as we will see later, is the basis of how separate pitched events are further grouped into phrases (or chunks).

2.3.3 Pitch intervals

A pitch interval is defined as the distance in pitch between two musical tones. In Western tonal music, an equal tempered twelve-tone chromatic scale, which is defined over a single octave, is employed. Using this system, octave equivalence in assumed. That is, tones separated by octave multiples are assumed to be musically identical. Furthermore, the octave is subdivided into twelve logarithmical equal steps (equal tempering). Each individual division of the octave is called a pitch-class, labelled as follows: C, C♯ (or D♭), D, D♯ (or E♭), E, F, F♯ (or G♭), G, G♯ (or A♭), A, A♯ (or B♭) and B. As a consequence of this division, thirteen unique pitch interval categories exist, including the unison and the octave. The distance between adjacent pitches on the chromatic scale is called a semitone and is therefore the smallest interval in Western music. Each interval is named in terms of the number of semitones between pitches. For example, the interval between C and G is 7 semitones. Alternatively, in music theory terms, an interval can have an additional label indicating its quality: unison, minor 2^{nd}, major 2^{nd}, minor 3^{rd}, major 3^{rd}, perfect 4^{th}, tritone (or augmented 4^{th}), perfect 5^{th}, minor 6^{th}, major 6^{th}, minor 7^{th}, major 7^{th} and the octave interval. This division of the octave into discrete categories is common to almost all musical cultures [13]. Given the relatively small number of discrete pitches that humans can perceive in STM, the division of the octave is considered to be related to the limits of the human auditory system.

Interval sizes can play a vital role in forming melody grouping boundaries. For example, pitch sequences containing relatively small interval distances (or steps) are usually grouped together. It is these steps that generally give the perception of motion in music. Sequences may also contain large intervals (or leaps); such leaps usually indicate group boundaries. It is important to note however that the distinction between a step and a leap is dependent on their context within a particular pitch sequence [151, pg. 129].

2.3.4 The octave interval

The octave interval possesses as special quality unlike any other interval. Pitches that are an interval of an octave apart, or have a frequency ratio of approximately 2:1, sound as if they are the same. The octave interval is a characteristic of virtually all musical cultures [13]. Consequently, although we use 88 individual pitches, there are only twelve distinct types of pitch-classes (or tone chroma), which repeat every octave. The same pitch category in different octaves is distinguished by specifying which octave it is in (C1, C2, C3, C4, C5, C6, C7 and C8). Each note therefore has two qualities; its octave number and tone chroma. Two distinct pitch heights may have identical pitch categories or, conversely, may have different pitch categories but be similar in pitch height. This relationship is very often represented using Shepherd's famous pitch helix [147]. Another interesting point about the octave is that the more an interval increases above the octave interval, the less the interval has any specific interval quality; it is instead interpreted as just a very large leap which, as we have already pointed out, is generally an indication of where a group boundary lies.

2.3.5 Tuning systems, scales and tonality

The construction of melodies is generally constrained by a tuning system, scales and tonality; all of which tend to be idiomatic of a particular musical culture. The tuning system of a particular culture generally consists of a specific number of subdivisions (or pitch categories) within an octave. Consequently, the pitch interval categories between such subdivisions are clearly defined. As we have already mentioned, Western music divides the octave interval into twelve distinct pitch categories giving rise to thirteen distinct interval categories (including the unison and the octave).

A musical scale defines a subset of intervals within a tuning system that may be used in constructing melodies. For example, Western music is generally constrained by the major and minor diatonic scales, which consist of seven notes. Other musical traditions use different scales; for example, folk music uses a pentatonic scale consisting of only five notes. One interesting point about scales relates to the number of notes each contains. For example, the major and minor scales consist of seven notes, which is commensurate with the STM span of humans.

The intervals that define a scale are generally not the same size. This inequality plays an important part in establishing the character of a scale and the centrality of a particular pitch. This gives us a sense of knowing where we are in the scale [149]. The central pitch of a scale is known as the tonic, and is generally referred to at important points of a melody, such as cadence points. Snyder [151, pg. 136] described tonality as *"a way of using a musical scale so that it seems to have a central pitch"*.

Tonality can be considered as a melodic schema which gives rise to expectations of what pitches will occur next. Consequently, our notion of a central pitch is learned through passive exposure within a particular cultural environment, without our even being aware that we are learning it. Therefore, tonality is a form of implicit memory. This is true whether a person is musically trained or not.

2.3.6 Melodic pitch contours

A melodic contour is characterized by the rising and falling motion of the pitch intervals in a melody. In non-tonal melodies, contour is the most memorable aspect [39]. This is a particularly important point with respect to the imperfect query experiments reported in Chapter 13, where queries may be so badly degraded that no sense of a central origin is present. In this circumstance, once the contour of a melody is preserved, melodies can still be recognized fairly accurately.

2.4 Rhythm

When two or more notes occur in sequence, within a relatively short time-span [27], the resulting pattern in time is called a rhythm. This section outlines some of the salient features of rhythm perception with respect to the work presented in this thesis.

2.4.1 Durations and inter-onset intervals

A note's *duration* is defined by the length of time which passes between a note's onset and its offset. On the other hand, an *inter-onset interval* (IOI) is the length of time which passes between the onsets of two consecutive notes. Regardless of where the offset occurs, a note always has the same attack point representing its onset. Consequently, it is the difference between successive onsets that is of most importance to rhythm perception. Therefore, in this thesis, IOIs are exclusively used to represent rhythms. An IOI with a duration of less than approximately 50ms is not perceived as two distinct event onsets but, instead, as a single event [151]. Furthermore, an IOI that is longer than approximately 2 seconds is also difficult to perceive. This is because the continuity between the two events is effectively lost. Consequently, the role of STM in rhythm perception is important.

2.4.2 Beat, pulse and tempo

A *beat* can be considered as an imaginary point in time with no duration. Beats are inferred directly from pieces of music and very often coincide with actual musical events; for example, from the attack points created by striking a percussion instrument. A melody's *pulse* is a sequence of isochronous (regular and recurring) beats which are equally spaced in time. The pulse of a melody is naturally inferred by humans (for example, tapping one's foot to music) and is common to virtually all music and all musical cultures. The *tempo* of a melody is the rate at which the pulse occurs (or the number of beats per unit of time) and is usually measured in beats per minute (bpm). Again, tempos can only be perceived if they occur within a particular range. If events occur too close in time they are not perceived separately. On the other hand, due to STM limitations, if they occur too far apart, they no longer seem connected.

2.4.3 Metre

Accents are associated with particular beats and give such beats a quality of weight or emphasis, which causes that beat to stand out with respect to other beats. Metre refers to the organization of beats into a recurring accent pattern. Accent information is provided by the nature of the event that occurs on a particular beat and can be established by grouping factors such as changes in intensity, duration or a leap in pitch. This gives rise to three types of accents, which act as perceptual cues for metre [84]:

Phenomenal accents. These represent physical events that occur within music. For example, the attack point of a note onset, sudden changes in dynamics or harmonic changes.

Structural accents. These relate to abstract musical concepts, such as the gravitational pull of tonal music towards the tonic.

Metrical accents. These correspond to beats that are perceived as relatively strong within their metrical context.

Metre is important in establishing the most salient temporal reference points in a melody; the points at which people tend to clap their hands or tap their foot. This has been referred to by Lerdahl and Jackendoff [84, pg. 21] as the tactus. Establishing the tactus-span allows us to recognize melodies in a tempo invariant fashion. In this thesis, IOIs are defined with respect to the tactus-span in order to achieve tempo invariance.

2.5 Memory and melody

When considering human melody perception, the manner in which our memory is organized, its limitations and how it functions are vital. In particular, our memory influences how we group sequences of musical events into chunks, which in turn influences how we can later recognize such groups as familiar. Memory for the auditory system, and indeed other perceptual processes, such as vision, may be loosely considered as consisting of three forms, each operating on distinct time scales: echoic memory, short-term memory (STM) and long term memory (LTM). Although these memory types are unlikely to operate independently of one another, the distinction is useful in describing how researchers generally believe memory functions. In this section, each of these types of memory will be discussed.

2.5.1 Echoic memory

Section 2.2 described how the acoustic vibrations in our surrounding environment are converted to electrical nerve impulses by our ears. This information is thought to persist in its raw continuous form in echoic memory for between approximately 250ms and several seconds [151, pg. 19] before it decays. It is believed that this information remains in echoic memory long enough to be processed.

This processing of the continuous steam of raw data which briefly persists in echoic memory is called feature extraction. This involves the extraction of features (by feature extractors) such as pitch, intensity, timbre and, equally important, the changes in these features over time. Features which occur simultaneously or are correlated in some way are perceptually bound into discrete coherent events or, in our case, notes of a melody. This feature extraction and perceptual binding has been referred to as *perceptual categorization* by Edelman [46]. In effect, this step, in the low level processing of musical events, reduces the vast amount of sensory information occurring in echoic memory to a relatively small number of discrete categories. After these features have been bound into discrete events, sequences of events are further grouped into chunks, based on perceptual cues such as proximity and repetition (see Section 2.7)

2.5.2 Short-term memory

Much of the structure we assemble from the world around us is a consequence of the limits of STM. Many researchers believe that, on average, the capacity of STM is seven *chunks* of information. In musical terms, a chunk may be considered as a single pitched event or a grouping of such events. Consequently, although we may store a relatively small number of chunks in STM at any given time, chunks themselves may contain sequences of items. For example, consider Table 2.1, our STM capacity is exceeded very quickly when trying to store this sequence. However, if we arrange these letters in the following way — *a chunk can consist of many items* — we can easily store the sequence in STM because it consists only of seven chunks.

a	c	h	u	n	k	c	a	n	c	o	n	s	i	s	t	o	f	m	a	n	y	i	t	e	m	s

Table 2.1: An example of chunking.

Although human STM can store approximately seven chunks, order information becomes confused for sequences containing more that about four chunks. This limit is known as the *transient memory span* (TMS) [114, pg. 77]. Typically, the maximum number of items a human groups together is consistent with the TMS.

STM is also limited in terms of its depth. That is, the length of time that information can be held in STM. Generally, it is believed that chunks can persist in STM for between approximately 3 to 5 seconds, after which they decay unless consciously rehearsed.

2.5.3 Long-term memory

LTM consists of knowledge about events which occurred in the past. They are not part of our conscious awareness unless invoked. LTM can be activated ether unconsciously, when we recognize or are reminded, or consciously, when we are recollecting. Elements of LTM usually take weeks or months to form and are usually created by simple passive exposure within our environment [154]. Memories of past experiences have a vital role to play in our perception of the present.

As we have pointed out, one fundamental characteristic of chunking is that chunks can become members of other chunks. Thus, our LTM consists of a hierarchical organization of chunks [36, pg. 366], where, as we progress up the hierarchy, we establish more abstract representations of our memories. This has implications for memory recall because it is harder for us to begin to recall from the middle of a chunk stored in LTM. Therefore, access to LTMs are primarily restricted to the boundaries between the chunks we form. This is because LTM is reconstructive, one event leads to or is associated with the next, and so the time order of events is crucial. Therefore, in constructing melodic queries, humans are unlikely to begin in the middle of a chunk. Instead, a memorable place, such as the beginning of a verse or a chorus, becomes the starting point. This is the primary reason why we believe melodies should be organized as a hierarchy of segmented melodic phrases within a CBR system.

2.5.4 Invariance in LTM for melodies

Levitin [85, pg. 214] described a melody as *"an auditory object that maintains its identity under certain transformations"*. For example, a melody can start at any note and it will still be recognized. This is known as transpositional invariance. Furthermore, melodies can be recognized regardless of how quickly or slowly they are presented (within limits). This ability is known as tempo invariance. Levitin [85, pg. 215] further describes why we can recognize melodies so robustly — *"One of the reasons we are able to recognize melodies is that the memory system has formed an abstract representation of the melody which is pitch-invariant, and so on"*. Consider the distinction between the following statements:

1. Melodies are *heard* as sequences of absolute pitches occurring at particular points in time.

2. Melodies are *stored* as an ordered set of relative pitches occurring at relative points in time.

Consequently, a melody retrieval system must be capable of recognizing a query as familiar, even if certain attributes of the original melody have been altered.

2.6 Stream segregation

Within our surrounding environment, we are constantly hearing sounds which emanate from a variety of different sound sources. For example, working in an office we hear the hum of an air conditioning system, phones ringing, distant conversation, a radio and so forth. The sound vibrations from these sources are continuous and do not necessarily have obviously defined divisible components. All of these sounds are converted from mechanical sound vibrations to nerve impulses by our ears. Consequently, unless mechanisms exist which segregate sound vibrations from different sound sources into individual streams, we would be unable to make sense of the sound landscape in which we live. This process is known as auditory stream segregation [12, 37, 148]. We group sounds that are similar in pitch, timbre, intensity and spatial location as sounds from a single coherent source, which is perceived as a single auditory object. In this thesis we are interested in processing a single stream; that of a melody.

2.7 Grouping mechanisms and melody

Snyder [151, pg. 34] points out that sounds *"may be grouped in either the melodic or rhythmic dimension, and usually grouping forces are operative in both these dimensions at the same time."* Furthermore, Krumhansl [76, pg. 153] states that *"two influences [grouping mechanisms] may simultaneously operate to converge on the same grouping. In other cases, two influences may tend toward different groupings, and one will predominate the other."* In the experiments described in Chapter 12 this is precisely what happens. Pitch and rhythm grouping cues at separate modules operate in isolation until a point is reached where stable groupings have formed at each dimension. At this point, the groupings at each module may agree or conflict. Where groupings conflict, the rhythm dimension will dominate. This section discusses some of the mechanisms involved in the grouping of melodies into chunks which, as we have discussed, are organized in a hierarchical fashion in LTM.

In addition to stream segregation, three types of grouping occur, which tend to coincide with the three forms of memory and time frames discussed in Section 2.5. Firstly, although a melody involves a continuous stream of acoustic vibrations, our brain organizes a melody into discrete events. The boundaries between distinct events are indicated by changes in frequency, loudness or timbre. Secondly, these discrete events are further organized into groups (or phrases or chunks), which can also be considered as a unit. This organization occurs due to pitch and rhythmic cues, such as similarity of interval (in time and space), where gaps often strongly indicate boundaries between groupings. Finally, larger groupings also occur, at the level of entire sections of music, where the general shape of an entire melody is established. The remainder of this section outlines some of the primitive grouping mechanisms that facilitate the formation of note groupings. Although other factors affecting the formation of group boundaries exist (such as the tonal aspects of a melody), we outline only those which are pertinent to this thesis.

Temporal proximity. Sequences of relatively short IOIs tend to be perceptually grouped together. Furthermore, if the IOI between two note onsets is relatively large, a grouping boundary is established at this point. This is known as the *gap principle* of rhythm perception. Of all the grouping factors, temporal proximity has the greatest influence over the formation of group boundaries [84].

Pitch proximity. Similar to temporal proximity, tones that are perceived as being close together in pitch tend to be perceptually grouped. Conversely, when the interval between two pitches is relatively large (a leap), this also indicates a group boundary.

Repetition. The repetition of phrases within a melody plays an important part in grouping. When a repetition occurs, it acts like a rehearsal mechanism, giving the phrase a more prominent role within the context of the entire melody. Lerdahl and Jackendoff [84] refer to this concept as parallelism and stress its importance as a grouping cue.

Primacy and recency effects. Studies in serial order have shown that items which begin and end a sequence are more likely to be recalled accurately than those which occur in the middle of a sequence. Indeed, cognitive models exist that try to capture this behaviour [67]. In Chapter 12 we show that SONNET-MAP exhibits such effects (particularly the primacy effect).

2.8 Summary

In this chapter, we discussed the aspects of melodic memory and perception that are pertinent to the work presented in this thesis. In particular, we outlined current thinking regarding how psychologists believe melodic memories are segmented into chunks using melodic grouping cues, and how these chunks are organized into hierarchies. This hierarchical approach to creating and storing melodies is particularly relevant to how the SONNET-MAP and *Re*TREE*ve* networks, introduced in Chapters 10 and 11, are applied to the task of CBR.

Chapter 3

Content-Based Retrieval of Melodies

3.1 Introduction

Content-Based Retrieval (CBR) of melodies is a sub-branch of the field of Music Information Retrieval (MIR). Although MIR is a rather new field of research, it has quickly expanded to include a very diverse range of algorithmic techniques from an equally diverse range of scientific disciplines. The multi-disciplined nature of MIR makes the field both interesting and challenging. In general terms, the CBR of melodies involves measuring the similarity of an input query (see Section 3.4), which consists of a melody fragment, against a database of stored melodies. The aim is to produce results containing a ranked list of melodies to which the fragment most likely belongs. The ability to recognize and retrieve melodies in this way has a variety of uses, both commercial and academic (see Section 3.2).

CBR systems exist that process audio file formats, raw acoustic signals or symbolically notated music. Hybrids of these systems also exist where, for example, audio signals are converted into symbolic notation, which is then used to search a symbolic database of melodies. Regarding these systems, a variety of techniques are utilized to retrieve melodies, although retrieval is predominantly accomplished by using a combination of string matching algorithms and indexing methods (see Sections 3.5).

In this thesis, psychologically-based symbolic melody representations are used to represent monophonic melodies (see Section 3.3). These representations are used as input to our SONNET-MAP and *Re*TREE*ve* systems (see Chapters 10 to 13). However, within these systems, melodies and queries are represented sub-symbolically by short-term cell activations and long-term connection strengths. Consequently, we presuppose the preliminary analysis of a melodic source (whether it be an audio file, or raw acoustic information) by a subsystem capable of extracting the required features of a melody (such as the pitch-height, IOI content, tonal centre and tempo). We therefore focus on how the information that is available at this level of abstraction can be utilized. At this level, there is a wealth of information from which to infer the structure of a melody (similar to the hierarchical structure of words, sentences and paragraphs) which can be effectively used during the retrieval process. To date, most CBR systems do not take into account this information to structure their data.

In this thesis, our approach is to organize and retrieve melodies using methods inspired by how humans might achieve this task (see Section 3.6). Therefore, the manner by which a human listener constructs mental representations of melody in STM and LTM, and how this information can be recognized at a later time, is paramount. This human-inspired approach to intelligent information retrieval is relatively new in the field of CBR.

3.2 Applications of content-based retrieval

The ability to recognize and retrieve melodies has a variety of uses, some of which are discussed here:

Query-based systems. These systems involve queries generated by human users. For example, a consumer may wish to locate a particular piece of music for which he or she has no textual information (for example, the melody's title or its composer). However, if the consumer can reproduce a fragment from the piece in question, the use of a CBR system may help the consumer to track down the music they have been looking for. Queries could be generated in a variety of ways. For example, by reproducing the chorus or a verse, which may be achieved by humming or playing playing a musical instrument. Alternatively, tailored computer software, such as a virtual piano keyboard, could be used to compose a query before it is submitted to a search engine. These applications would be particularly useful for record companies trying to boost sales; whether on the high street or over the internet.

Musicology. Musicologists, who are interested in studying melodic similarity, may find CBR systems useful. Perhaps in establishing the resemblance between different musical styles; for example, how sixties or punk music influenced more contemporary music. Traditionally, musicologists study similarity by analysing recordings and music scores. The automation of this process could aid in this process and yield insights not previously considered.

Copyright and royalty issues. For music rights organizations, this technology can aid in protecting artists from plagiarism, and consequently, protecting an artist's royalty income. Conversely, artists could ensure that they are not exposing themselves to the risk of being accused of plagiarism.

Recommendation systems. Online record companies could build up a profile of a consumer's musical tastes by analysing the content of previous purchases. Recommendations of similar music could then be made [26, 89].

3.3 Melody representations

There are variety of possible approaches to representing music in CBR systems. Although popular file formats (such as WAV, MP3, WMA and MIDI) can be utilized, this thesis is primarily concerned with psychology-based symbolic representations and neural-based sub-symbolic representations. Symbolic representations can be derived directly from acoustic signals or melodic file formats. Sub-symbolic representations in ANNs are generally derived from the presentation of symbolic information. Traditionally, MIR systems represent both queries and the stored melodies as strings of symbols. In contrast, using the SONNET-MAP and *Re*TREE*ve* networks introduced in this thesis, these symbolic representations are only used as melodic input. Within the networks, melodies are represented sub-symbolically as short-term and long-term memory traces.

The choice of which representation to use has important implications for how melodies can be retrieved and, in particular, how accurately they can be retrieved. For example, in Chapter 13 we show that certain melody representations produce better retrieval results in one context but not in others. Furthermore, representation usually dictates the algorithmic approach to matching queries against a melodic database. Moreover, the representation of melodies in retrieval systems should have the properties of transpositional and tempo invariance. This enables queries to be recognized, regardless of the key in which a query is produced or the tempo at which it is presented.

In humans, pitch-height representations are formed through the process of pitch perception, where the spectral components of an acoustic vibration are fused into the perception of a single pitched event (see Chapter 2). Pitch-heights can also be extracted directly from audio files using pitch extraction techniques [48, 117, 172]. Pitch-heights are not suitable representations for the direct comparison of queries against a database of melodies, because transpositional invariance is not accounted for. In a pitch-class representation, where pitch-height is collapsed across all octaves, pitch-heights are still preserved within the octave. Consequently, this representation is not suitable either. Nevertheless, pitch-heights can be used to derive other representations, which do possess the property of transpositional invariance. In this thesis, we use invariant pitch categories (IPCs), pitch-intervals and contours to represent the pitch facets of melodies, each of which accounts for transpositional invariance.

Like pitch information, rhythmic information (such as beat, tempo and onset times) can be extracted directly from raw acoustic signals or audio files. This is achieved using rhythm extraction techniques, which have recently been of much interest [1, 29, 32, 33, 45, 52, 97, 120, 123, 128, 161]. In this thesis, IOIs are used to represent the rhythmic facet of a melody. They are specified in terms of a melody's tactus, thus tempo invariance is accounted for.

Generally one, or a combination of, these representations are used by most retrieval systems. The remainder of this section describes each of these representations. Refer to Figure 3.1 and Table 3.1 for examples of each. Note that the tactus-span associated with this melody fragment is 620ms (= ♩).

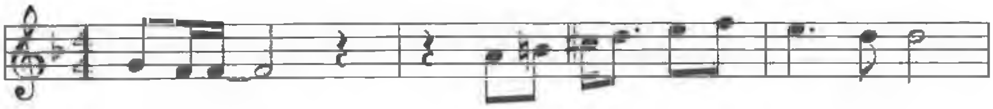


Figure 3.1: Music notation of the first three bars of the melody *Yesterday* (see Section E.4.1). Refer to Table 3.1 for the corresponding IPC, pitch-interval, melodic contour and IOI representations.

IPC	2	0	0	4	6	8	9	11	0	11	9	9
Pitch-interval	—	2	0	4	2	2	1	2	1	1	2	0
Contour	—	D	S	U	U	U	U	U	U	D	D	S
IOI	—	0.5	0.25	4.25	0.5	0.5	0.25	0.75	0.5	0.5	1.5	0.5

Table 3.1: The IPC, pitch-interval, contour and IOI representations for the melody fragment presented in Figure 3.1. Remember, for the IPC representation, since each category is normalized with respect to a tonal centre (to account for transpositional invariance), the key of the piece must be known. Moreover, since IOIs are represented as percentages of the tactus-span (to account for tempo invariance), the presentation rate of a query must also be known.

Interval representation. Pitch intervals represent the difference in pitch between adjacent tones, and have been widely used in melody retrieval systems. One disadvantage of using a pitch-interval representation arises due to the exacerbation of local errors (see Section 3.4).

Invariant pitch-class representation. With the IPC representation, pitch-heights are collapsed across all octaves. This reduces the pitch range to a single octave containing pitch-classes; thus accounting for octave equivalence. However, as we have already pointed out, since absolute pitch levels within this collapsed octave are preserved, invariance under transposition is not accounted for. This problem is addressed by normalizing each sequence with reference to a tonal centre, or key. In the case of Western tonal music employing equal-tempered tuning, each pitch is represented by one of twelve chromatic pitch categories, depending on a melody's tonal centre. These categories are labelled from 0–11. For example, the *tonic* will always be represented by 0, regardless of the melody's absolute pitch.

Melodic contour. Most retrieval systems to date have utilized the melodic contour representation for representing melodies in databases. Three contour categories exist; U (up), D (down) and S (same), where U represents an increasing interval, D represents a decreasing interval and S represents a repetition of the same pitch. This representation is advantageous for melody retrieval because, even if the pitches of a query are badly degraded, recognition remains robust, provided that the contour of the original melody is preserved. However, this representation is disadvantageous with respect to the generality in which melodies are represented. This is because there is not enough discriminatory information provided to distinguish between similar but distinctly different target melodies. Consequently, for queries which are fairly representative of the target melody, the pitch-interval and IPC representations are more suitable.

Rhythm. An IOI represents the time interval between the onsets of consecutive notes. Tempo invariance is achieved by defining IOIs in terms of a melody's tempo (or tactus-span).

To see how each of these representations can be embodied in a SONNET network as STM and LTM, refer to Chapter 9.

3.4 Melodic queries

By naming a familiar melody, humans can recollect and perhaps sing the melody. The problem to be addressed in this thesis is precisely the opposite. Instead, given a melodic segment, the task is to recognize and name the melody from which the fragment most likely originated. It is this melodic fragment that is known as the *query*. In other words, *a query is a melodic fragment, of any length, that is used as a cue to retrieving the target melody.* A query may originate from anywhere in the melody, not necessarily from the beginning. A query may be an exact copy of a segment from the original melody or it may be an inaccurate copy of a segment from the original. Consequently, its rhythm may be incorrect, its pitches may be incorrect, some notes may be missing or some notes may be added. In Chapter 13 we automatically generate queries taking these types of errors into account. However, we do not claim that these distorted queries perfectly model human performance at query generation, rather that it models the types of errors humans make [100]. A query may take any one, or a combination, of the following forms.

3.4.1 Perfect queries

A perfect query is an exact copy, or reproduction, of a segment from the target melody. The target melody is the melody intended to be retrieved. Consequently, perfect queries will tend to retrieve the target melody with most accuracy.

3.4.2 Imperfect queries

An imperfect query is an inaccurate reproduction of a segment from the target melody. Consequently, it is more difficult to retrieve target melodies using imperfect queries. Imperfect queries may take many forms depending on the type of errors they contain.

Noisy queries. This type of query can be described as a *noisy* version of a perfect query. A noisy query is typical of the type of query one would expect a human singer to produce. Some (or all) of the notes (pitch and/or rhythm) may be incorrect to varying degrees depending on the memory or skill of the person who creates the query. The higher the noise level the more challenging the problem. Generally, errors are local, for example, using the pitch-interval representation, if a note's pitch happens to be inaccurate, the interval from the preceding pitch to the inaccurate pitch will be wrong. If the subsequent pitch is specified accurately, the interval from the incorrect pitch to the subsequent pitch will also be wrong. Therefore two errors have actually occurred — this is called a local error.

Queries with insertions and deletions. Finally, it is common to produce queries that contain extra notes (insertions) and/or missing notes (deletions). These type of errors are the most difficult to deal with.

3.5 Other retrieval systems

As we have just pointed out, most CBR research represents a melody (both the query and the database entries) as a string of symbols that are drawn from a symbolic alphabet representing melodic features. Consequently, the retrieval process is reduced to the task of string matching, which is utilized to retrieve the ranked list of potential matches. Typically, for perfect queries, standard string matching algorithms such as Knuth-Morris-Pratt and Boyer-Moore can be used [144].

	Input		Database			Queries		Melodic Features				References
Name	Audio	Symbolic	Audio	Symbolic	Sub-symbolic	Perfect	Imperfect	Pitch	Rhythm	Interval	Contour	
MELDEX	●	●		●		●	●		●	●	●	[4, 98, 99]
QBH	●			●		●	●				●	[55]
Themefinder		●		●		●		●		●	●	[75]
C-Brahms		●		●		●	●	●	●	●		[82, 83, 163]
Musipedia	●	●		●		●	●				●	[130]
GUIDO/MIR		●		●		●	●	●	●	●		[68]
CubyHum	●			●		●	●		●	●		[122]
*Re*TREE*ve*		●			●	●	●	●	●	●	●	[65]

Table 3.2: A comparison of some well-known melody retrieval systems with *Re*TREE*ve*.

For imperfect queries, approximate string matching must be performed. This is achieved through dynamic programming, where editing distances are computed between a query and the melodies in a database. This process in facilitated by the use of an indexing method (such as inverted files, B-trees or signature files). To accommodate the so called *"lack of the equivalent words in music"*, the use of the n-gram style approach, which breaks up a melody into n-sized chunks, can be used for indexing [40].

Over the past decade, the number of melody retrieval systems described in literature has grown significantly. It has become somewhat cumbersome to describe each and every one. Instead, some of the better known melody retrieval systems are listed in Table 3.2, in conjunction with some of the major characteristics of these systems. For more detailed information, see the following references: Blackburn and Rouree [8, Chapter 2], Downie [42], Sødring [152, Chapter 4], and in particular Typke et al. [162].

3.6 A human-inspired approach

Intelligent human behaviour can be characterized by its autonomy, adaptability, robustness and real-time operation within unconstrained, unpredictable, complex and dynamic environments. In particular, human listeners are capable of spontaneously creating and organizing melodic memories, and subsequently recognizing these melodies. When recalling a melody, a listener cannot usually begin at an arbitrary point in the melody. Instead a memorable place, such as the beginning of a verse or a chorus, becomes the starting point. Studies in music psychology suggests that these memorable points (or boundary points) are determined using perceptual grouping cues. The phrase segments that form between consecutive boundary points are further organized in a hierarchical fashion, which facilitates the learning of entire melodies. Furthermore, familiar melodies can be robustly recognized at a later time, even if these melodies are not accurate reproductions of the originals. These abilities make human listeners ideal for the task of CBR. Research into these abilities suggest possible algorithmic and structural approaches to CBR research.

The use of the n-gram style approach to indexing breaks a melody into n-sized chunks for indexing. These methods assume that there are no equivalent structures, such as words and sentences, in melodies. However, as we have just pointed out, melodies are in fact highly structured. Consequently, such approaches fail to acknowledge the inherent phrased structure of melodies and the wealth of melodic information available to infer such a structure. Nevertheless, recently, researchers have begun to acknowledge the importance of this information and, consequently, many have begun to investigate melodic similarity [3, 108, 121] and segmentation techniques [79, 101, 109, 124] for structuring melodic information in retrieval systems.

This dissertation argues the case for intelligent melody retrieval that is inspired by research which suggests how humans achieve this behaviour. In this thesis, we introduce a neural network known as SONNET-MAP, which segments melodies into chunks of varying sizes based on perceptual cues such as proximity and repetition (see Chapters 10 and 12). These chunks are then used to populate the *Re*TREE*ve* network, which is derived from SONNET-MAP (see Chapters 11 and 13). *Re*TREE*ve* organizes melodies in LTM as parallel hierarchies of automatically segmented melodic phrases, which are matched against during retrieval.

3.7 Summary

This chapter outlined the problem of content-based melody retrieval. Issues such as application, representation and query errors were discussed. In particular, we outlined our human-inspired approach to solving the problem. The following chapter outlines the use of ANNs for music processing, and makes the case for using SONNET-based networks for melody retrieval.

Chapter 4

Artificial Neural Networks and Music

4.1 Introduction

In recent years, the application of artificial neural networks (ANNs) to the perception of music has become increasingly dominant over traditional symbolic approaches. ANNs are inspired by their biological counterparts in the human brain and comply with our human-inspired approach to melody retrieval, which we have outlined in Chapters 1 to 3. This chapter discusses the foundations of ANNs in neuroscience and discusses the reasons behind why ANNs have been increasingly chosen to model aspects of music perception. Moreover recent ANN models, which have been applied to music processing, will be described. Finally, the primary motives behind our choice of using SONNET-based research throughout this thesis is reviewed.

4.2 Foundations in neuroscience

The structure and operation of ANNs are inspired by their biological counterparts in the human brain. Although current understanding of the brain is limited, a fair amount is known about individual nerve cells (or neurons) with respect to their anatomy, physiology and how they communicate. In general, neurons consist of three major components: the dendritic tree, the cell body (or soma) and an axon. Although each component has a specific function, they each complement the overall operation of a neuron.

The dendritic tree is the input mechanism of a neuron. Electrical impulses, from the axons of other neurons, cause the release of neurotransmitters at points called synapses. A narrow gap, called the synaptic cleft, separates an axon from a dentritic branch, and facilitates interneuron communication by transporting the released neurotransmitters to receptor sites on the dentritic branch. Many types of neurotransmitters exist; some have an excitatory effect and tend to cause neurons to activate, whereas others have an inhibitory effect and tend to suppress the activation of a neuron. The soma combines all of the signals received by its dendritic tree, producing a membrane potential. At this point, if the the soma's membrane potential is sufficiently large, the cell becomes active, causing an electrical pulse to be transported down the axon (a neuron's output mechanism) to other neurons. The strength of synaptic connections can be varied which, over time, can increase or reduce the influence one neuron has on another. This is the general principle by which learning occurs in the brain.

The brain consists of an enormous number and variety of neurons (one estimate is approximately 100 billion, Nolte [115, pgs 2–3]) which are massively connected. Consequently, although the operation of each cell is relatively slow when compared to modern digital sequential processors, it is the parallelism at which neurons operate that empowers them collectively to perform tasks efficiently and intelligently. Indeed, James McClelland writes *"If current estimates of the number of [connections] in the brain are anywhere near correct, it would take even the fastest of today's computers something between several years and several centuries to simulate the processing that can take place within the human brain in one second"* [30, pgs 187–188].

The artificial counterparts of biological neurons are known by many names; cells, nodes and units for example. Generally, in this thesis we will refer to an artificial neuron as a cell. The basic components of an artificial neuron and its function are analogous to the basic components of a biological neuron (see Figure 4.1). An artificial neuron consists of input connections (dendrites), the main processing unit (soma) and the output activation (axon). The combined input to a cell is typically summed in order to produce the output activation, which is propagated to other cells. Synapses are modelled by weights which represent the strength of a connection. Like biological synapses, the strength of a weight can be modified. This is achieved using a learning algorithm, which usually utilizes the cell's activity and a learning rate parameter, in evaluating the extent of a weight change. The strength of all the adjustable weights in a network encode the knowledge learned by that network. Cells can be connected to each other in a variety of ways (see Figure 4.2 for one example). In conjunction with the topological configuration of a network, ANNs can be generally categorized by the type of learning they implement:

Supervised learning. With supervised learning, a set of training pairs, each consisting of an input vector (the external environment) and its desired output vector (the supervisor) are presented cyclically to the network for a prescribed number epochs (or training cycles). By comparing the actual output vector with the desired output vector, an error vector is produced, which is fed back through the network. The learning algorithm adjusts the weights such that the error between the actual output and the desired output is minimized. Once the network has been trained to produce the desired output in response to each input vector, the network should be able to generalize and produce suitable outputs in response to novel input vectors.

Reinforcement learning. With reinforcement learning, only the input vector is presented to the network. In response to output which is unsatisfactory, the supervisor indicates to the

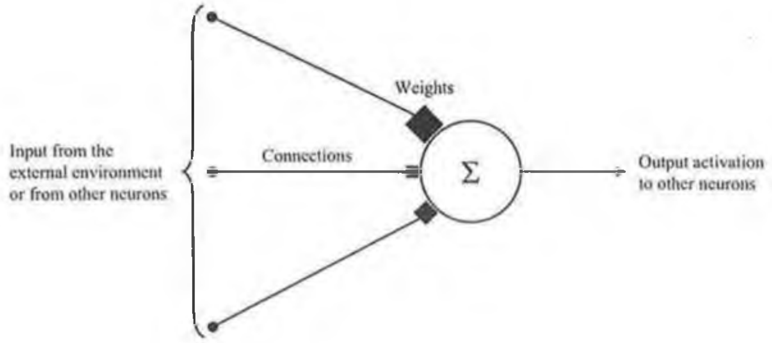

Weights
Input from the external environment or from other neurons
Connections
Σ
Output activation to other neurons

Figure 4.1: An artificial neuron.

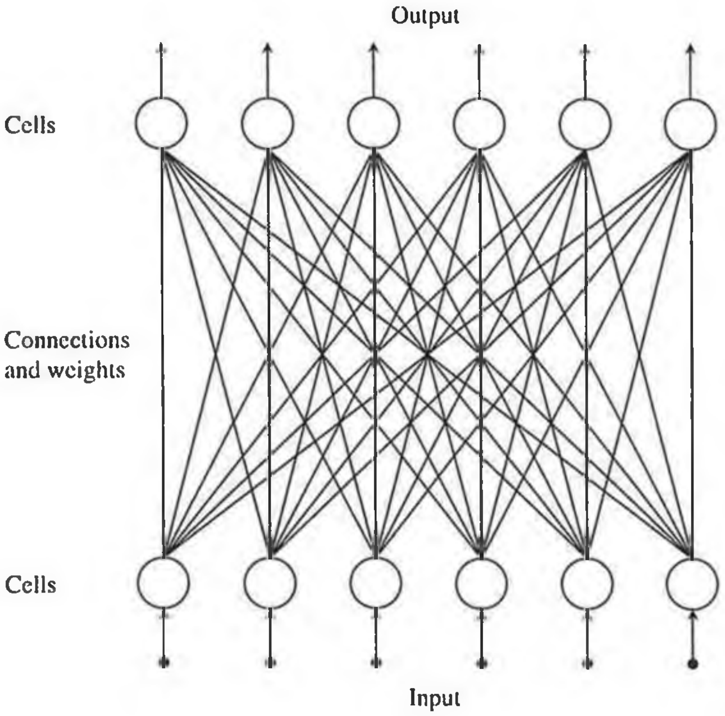


Figure 4.2: Artificial neurons are connected to form a network.

network that it has made a mistake. However, the desired output is *not* provided to assist the learning algorithm in adjusting the network's weights.

Unsupervised learning. With unsupervised learning no supervisor is utilized. Consequently, these networks operate autonomously, and adjust their weights such that meaningful patterns, which are present within an input vector, are learned. It is this type of learning which forms the foundation of the SONNET-based networks utilized in this thesis.

Although no researcher would claim that ANNs accurately model the architecture and operation of biological neural mechanisms, many researchers do try to accurately capture a variety of well established principles known to exist within these biological systems. Their goal is to achieve evolutionary, biologically and psychologically plausible models of human cognition. While these goals are considered important in the context of this thesis, we will not be constrained by them to the detriment of building a useable CBR system for melodies. For more details on the role played by neurons in brain function and on their artificial counterparts, refer to the following sources: [2, Chapters 1 and 2], [34, Chapter 1], [66, Chapter 1], [170, Appendix A] and [74, 115, 138, 154].

4.3 Why Artificial Neural Networks and music?

Early AI research was dominated by the tenet that the mind is a symbol system whereby human thought involves the manipulation of symbols prescribed by formal rules. This hypothesis, known as the *symbol system hypothesis*, forms the basis of symbolic AI and the foundation of early models of music cognition. Although symbolic-based approaches have achieved some success in this area, there are growing concerns among music theorists regarding certain drawbacks that are inherent in this approach [38]. Typical rule-based music research (and indeed symbolic AI) has been necessarily restricted to simple, predictable and relatively static toy musical environments. Systems developed using rule-based technology have difficulty generalizing knowledge from their input. Consequently, domain specific heuristics are hard-coded to aid in problem solving. The manner by which these heuristics, and indeed knowledge in general, can be autonomously learned presents another significant drawback. In other words, rule-based symbol systems have difficulty in adequately addressing the *knowledge problem* [30]. Indeed, Bharucha and Todd [7, pg. 130] write:

> *"Although rule-based models of music have been successfull at describing the formal structure of some musical compositions, and have thus provided valuable hypothesis and analytic constraints, they fall short as psychological theories. They fail to account for the acquisition of the rules they postulate, and this ad hoc postulation of rules is not typically limited to a small set of assumptions of which the others are a natural consequence."*

Furthermore, these systems tend to have difficulty with noisy input environments, which in music, are commonplace. These shortcomings are serious if we intend to model the real-time, autonomous, adaptable and robust behaviour of a human listener.

ANNs offer novel insights and have the potential to address these issues [96]. In particular, ANNs have some or all of the following properties, which are desirable for modelling aspects of human musical behaviour:

Learn from exposure. ANNs learn by receiving and responding to input patterns which model a particular input environment. The knowledge they encode is created solely on the basis of regularities inherent within this environment. No formal rules or domain specific heuristics need be applied. Indeed, Franklin [51] suggests that this ability to extract information present within an environment is one vital aspect in developing plausible models of the human mind. In Chapter 12 we show how SONNET-MAP can autonomously learn significant patterns within a set of melodies.

Spontaneous generalization. Given the knowledge encoded by an ANN, when patterns the network has already learned are presented in different contexts, ANNs generally perform well in categorizing these patterns as familiar.

Graceful degradation. Given degraded (fuzzy) input, an ANN's performance at pattern recognition degrades gradually as noise levels increase. This ability to operate under the presence of noise is of particular importance for melody retrieval tasks and is one major motivation behind our choice of an ANN approach to the problem of melody retrieval (see Chapter 13).

Content addressability. To retrieve a particular pattern stored in an ANN, only a portion of it needs to be specified. The network automatically finds the best match. This is precisely the property of ANNs we exploit in this thesis.

4.4 Applications in music processing

A wide variety of ANNs have been applied within the musical domain. In this section we briefly describe some of the most notable ANNs that have been used by music researchers, problems with these networks and a summary of their applications.

4.4.1 Feed-forward networks

Back-propagation networks

Arguably, the most widely used ANN to date has been the back-propagation network (BPN). BPNs generally consist of multiple layers of processing cells and are trained using the supervised back-propagation of error algorithm [139]. The back-propagation algorithm was responsible for the revival of interest in ANN research during the mid 1980's, particularly because of its strong mathematical foundation. The principal strength of the back-propagation of error algorithm is its general pattern-mapping ability, which provides a tremendous scope of applicability. In addition to its application in music processing, it has been applied to image compression [31], character recognition [80] and, most famously, speech production [145].

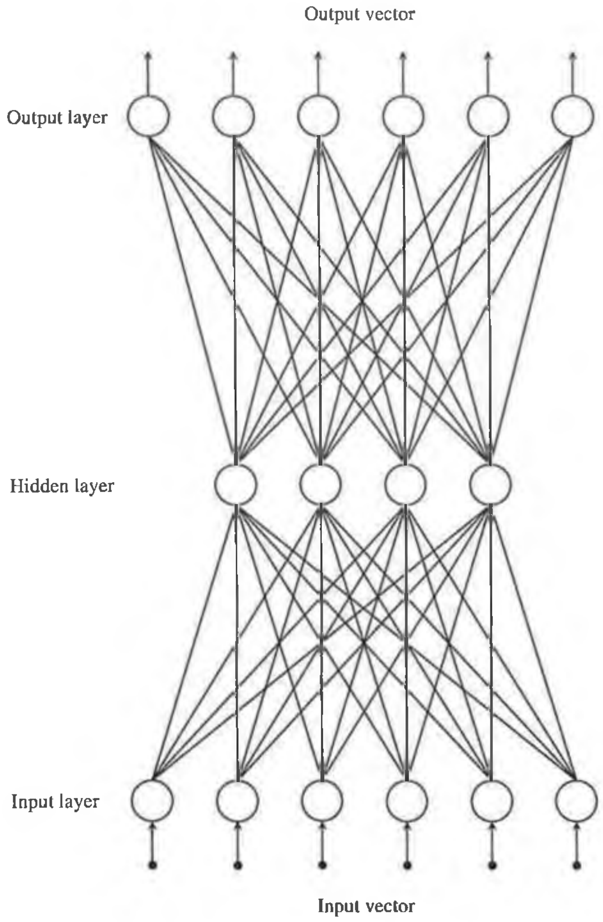


Figure 4.3: Multi-layered feed-forward network which is trained using the back-propagation of error algorithm.

Typically, a BPN contains three processing layers: the input layer, the hidden layer and the output layer (see Figure 4.3). The input layer is fully connected to the hidden layer. Likewise, the hidden layer is fully connected to the output layer. External input vectors (representing the external environment) are applied to the input layer. These signals are propagated to the hidden layer via adjustable connection weights. The hidden layer can be considered as a set of feature detectors, which organize through learning to represent the most salient aspects of the input vectors. The activations of the hidden layer are further propagated, via adjustable weights, to the output layer. Learning takes place such that the network learns to map the set of input vectors to their corresponding output vectors. Then, in response to novel patterns, the network should be able to generalize and produce suitable output in responses to novel input vectors.

One example, which applies a BPN to music processing, was presented by Laden and Keefe [77]. Their experiments examined the performance of a BPN in classifying chords as major, minor or diminished triads. A variety of different pitch representations were used as inputs to their system. They found that, for a BPN, psychoacoustic representations of chords (an harmonic complex representation and a subharmonic complex representation) performed better than the use of a pitch-class representation. Furthermore, the network was able to generalize and recognize some of the chords even when harmonic components were missing, or if inverted versions of the chords were presented.

Back-propagation networks with short-term memory structures

Back-propagation can also be applied to temporal sequence processing by utilizing a STM mechanism, such as a *tapped delay-line*, an *exponential trace memory*, or a *gamma memory* [106]. Of these, the tapped delay-line has been the most widely used (see Figure 4.4). Using this approach, when an input item is presented, prior items are shifted to adjacent elements, while the current item is presented to the first element [$x(t)$]. This causes the oldest item to be discarded. Typically, tapped delay-line memories are incorporated into feed-forward networks in one of two ways. The first possibility is to focus a tapped delay-line at the beginning of the network; this is called a *focused time-lagged feed-forward network* (focused TLFN). The other possibility is to uniformly distribute the tapped delay-line throughout the network; this is called a *distributed time-lagged feed-forward network* (distributed TLFN), although it has also been called a *finite impulse response* (FIR) network or a *time-delay neural network* (TDNN) [66]. The focused TLFN can be trained with the basic back-propagation algorithm, while the distributed TLFN must use the *temporal back-propagation* algorithm.

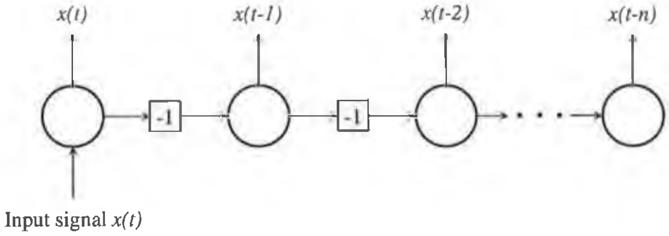


Figure 4.4: A tapped delay-line memory.

These networks have been successfully used for a variety of applications; most notably, for phoneme recognition [146, 166] and time series prediction [106, 167]. However, we are not aware of many applications of these networks to music processing. One system we are aware of, known as SONIC, which was introduced by Marolt [91], transcribes polyphonic piano music using a combination of adaptive oscillator networks and a time-delay neural network.

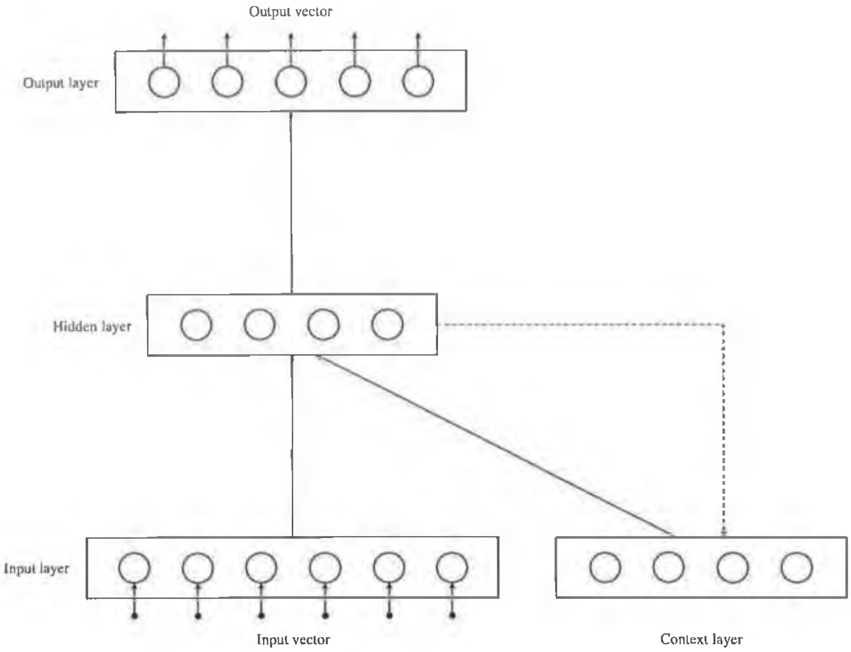


Figure 4.5: The architecture of a SRN known as the Elman network.

4.4.2 Recurrent networks

Time can also be incorporated into an ANN by the use of recurrent links. These ANNs can be either partially recurrent or fully recurrent, both of which are discussed in this section. *Simple recurrent networks* (SRNs), or *sequential networks*, consist mainly of feed-forward connections, with a carefully chosen set of time-delayed feedback connections. These feedback connections can emanate from the hidden layer (Elman [50] — see Figure 4.5) or from the output layer (Jordan [70]) and fed back to a layer of *context units.* SRNs can be trained with the conventional back-propagation algorithm provided that the recurrent connections are not adjustable. Typically though, the *temporal back-propagation* algorithm is used, which is an extension of the standard back-propagation algorithm.

SRNs have been a popular choice for processing musical sequences. Bharucha and Todd [7] show how a Jordan network can be utilized as a sequential memory for melodies, where both veridical expectations (the *instance-based* expectation for the next note in a familiar melody) and schematic expectations (*culture-based* expectations for notes that typically occur in familiar contexts) were generated. Additionally, Todd [159] and Mozer [105, 107] have shown how SRNs can be utilized to compose melodies. This is achieved by training the network with a selection of melodies and then letting the network produce output on its own (within certain constraints) given a starting note. In effect, previously learned melodic structures are used to produce novel compositions.

With a fully recurrent ANN, any cell can be connected to any other cell. These architectures can be trained using the *back-propagation through time* (BPTT) algorithm, where the network is unfolded through time into a layered feed-forward network, with an additional layer being added at every time step [176]. Alternatively, the *real-time recurrent learning* (RTRL) algorithm can be utilized [174, 175, 176]. To date, as far as we are aware, fully recurrent networks have not been applied much to music processing. However, one example by Dolson [38] shows how rhythmic sequences can be learned using a fully recurrent ANN trained with the RTRL algorithm.

4.4.3 Problems with gradient descent learning algorithms

All of the ANNs described thus far use gradient descent learning algorithms. By considering the error landscape in weight space, a number of features of this landscape are visible. The error surface tends to have hills, valleys, folds and gullies in multi-dimensional space. The goal of gradient descent is to find the lowest point in this landscape, which is called the global minimum, and avoid getting stuck in areas called local minima (the valleys and gullies). Despite the success of gradient descent, a variety of well known problems exist with this learning technique, which result in long, uncertain training procedures (see [2, Chapter 9], [34, Chapter 4], [66, Chapter 4] and [170, Chapter 3]). These problems severely limit the use of these networks given the way we intend to apply network architectures and learning in this thesis. This section summarizes these issues.

Scalability. To ensure stability, the step size (or learning rate) must be fairly small; thus convergence and run times are slow. Consequently, gradient descent requires hundreds, thousands or even tens of thousands of epochs for the network to achieve convergence for even the simplest of problems. Moreover, as the size of the network increases so too does the number of epochs required. Although techniques have been developed to reduce these effects, such as the use of a *momentum* term during learning [139], scalability is still a serious problem.

Local minima. The presence of local minima on the error surface means that it is possible for gradient descent algorithms to get stuck within these minima, even if deeper minima exist nearby. Methods do exist which can prevent a gradient-based algorithms from becoming trapped in local minima, but these are slow, which further exacerbates the scaling problem exhibited by gradient descent methods.

Catastrophic forgetting. Suppose a BPN is taught with a particular set of training pairs. Furthermore, suppose at a later time an additional training pair needs to be learned. In this situation, the encoding of previous information may be drastically degraded. Consequently, in general, if these networks are exposed to a continually changing input environment, where the same input vector may be seen for only a relatively short period, the training process may never converge. In other words, gradient descent learning does not solve the stability-plasticity dilemma which, for modelling human perception, is a necessity.

Optimizing the number of hidden cells. Generally, the number of hidden cells required in the hidden layer has to be optimized by trial and error. If too few cells are utilized, the training set may not be adequately learned. If too many cells are utilized, the network may not generalize appropriately. This problem is exacerbated if multiple hidden layers are utilized.

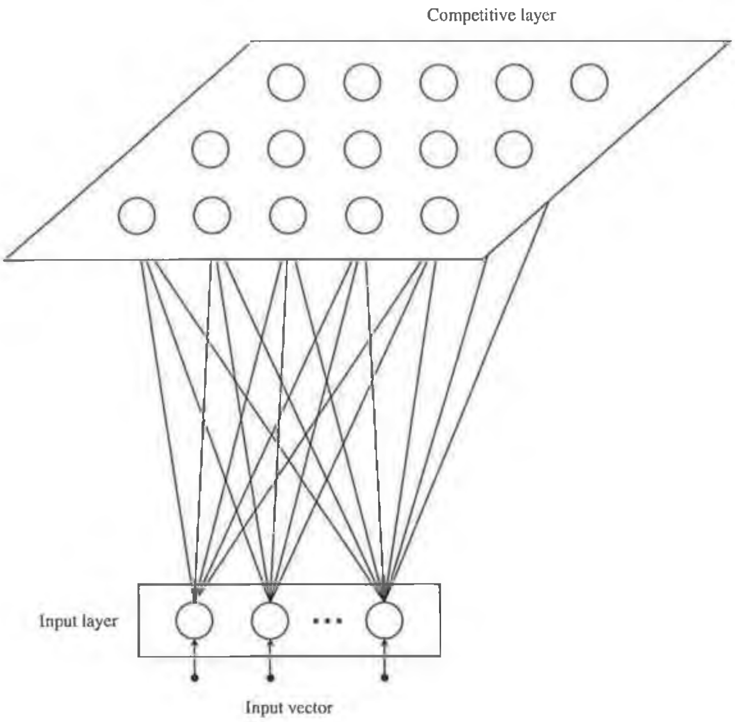


Figure 4.6: Kohonen self-organizing feature map.

Network paralysis. As training progresses, the values of weights can become very large, which in turn can cause the output signals of all or most cells to become large and, as Wasserman [170, pg. 57] points out, this can occur *"in a region where the derivative of the squashing function is very small"* and since *"the error sent back for training is proportional to this derivative, the training process can come to a virtual standstill"*.

Another issue which many biological purists quote is the lack of biological plausibility inherent in the gradient descent method. Irrespective of whether gradient descent is biologically plausible or not, the problems listed above mean that it is not a suitable model of human musical perception. The remainder of this section discusses a class of ANNs which use unsupervised learning and, consequently, through which many of the issues discussed here are resolved.

4.4.4 Self-organizing feature maps

Section 2.2.3 pointed out that some underlying principle governing brain plasticity is involved in the mapping of sensory information from its raw stimulus to its corresponding areas of the cerebral cortex. The development of self-organizing feature maps was motivated by this distinct feature of the human brain. Self-Organizing Maps (SOMs) are unsupervised and competitive networks, consisting of two layers which are fully connected: the input layer and a competitive layer (which is typically organized as a two-dimensional grid) [72]. During learning, SOMs modify their weights to reflect statistical similarities inherent in the input vectors presented to it (see Figure 4.6).

Kohonen [71] describes how such a network can be utilized to convert spoken language into typed text. With regard to music processing, such a network can also be used. In particular, Leman [81] described results which show how a SOM organizes itself in accordance with the circle of fifths when presented with a set of triad and seventh chords (using a representation based on Terhardt's theory of pitch extraction [157]). We consider the SOM network important in explaining how the tonotopic organization of the shared event field presented in Chapter 9 could have emerged.

4.4.5 ART

Section 4.4.3 explained that gradient-based learning algorithms fail to solve the stability-plasticity dilemma. Consequently, if these networks are exposed to a continually changing input environment, the network may never learn any salient aspect of this environment. To address this issue, a class of ANNs, collectively known as *Adaptive Resonance Theory* (ART), have been invented (ART 1 [15, 61, 62], ART 2 [14] and ART 3 [16, 17]).

An ART network consists of an input field, $\mathcal{F}_1$ (which receives input from $\mathcal{F}_0$) and a classification field, $\mathcal{F}_2$ (see Figure 4.7 for an example of an ART 1 network). Feed-forward connections fully connect $\mathcal{F}_1$ to $\mathcal{F}_2$. Additionally, feedback connections fully connect $\mathcal{F}_2$ to $\mathcal{F}_1$. The classification field implements on-centre off-surround competition. That is, each cell sends an excitatory signal to itself (on-centre) and an inhibitory signal to every other cell within $\mathcal{F}_2$ (off-surround). Classification cells compete with each other to learn and classify input patterns across $\mathcal{F}_1$. When an input pattern is presented, competition yields a winning cell at $\mathcal{F}_2$. Then, an orienting subsystem matches the input vector against the feedback vector produced by the winner. If the match is within the specified vigilance, adaptive resonance is said to occur. This results in the bottom-up and top-down weights of the winning cell being adjusted to make them more like the input pattern. If the match is insufficient, the current winner is temporarily disabled and another cell is chosen. This process continues until a suitable winner is chosen, whose pattern matches the input within the specified vigilance. If no suitable cell can be found, an un-instantiated cell is chosen to learn the pattern. After learning, this cell is said to be committed to the pattern. No committed pattern is ever modified unless it matches an input pattern within the limits specified by the vigilance parameter. In this way, the stability-plasticity dilemma is resolved.

Due to their ability to solve the stability-plasticity dilemma, ART networks have become particularly popular within music research. One notable example by Griffith [59] illustrates how an ART 2 network can be integrated into a system of ANNs that shows *"how patterns of pitch and interval can be used to induce exemplars of tonal centres and abstract pitch"*.

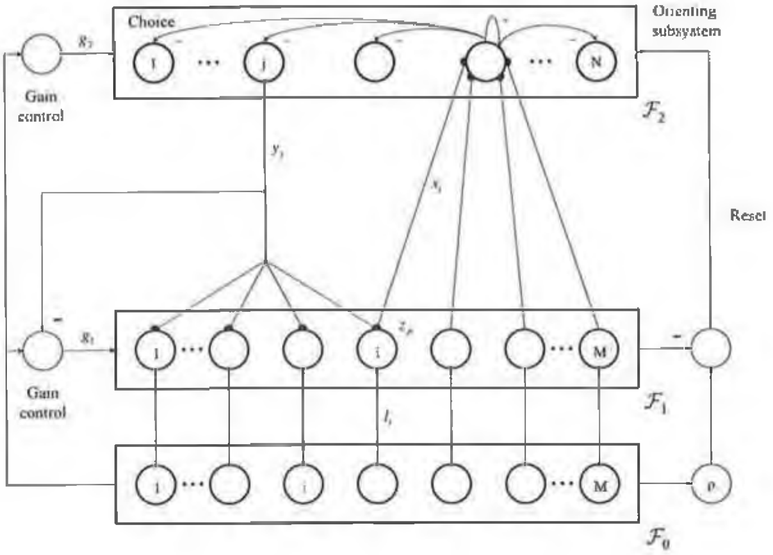


Figure 4.7: The architecture of an ART 1 network.

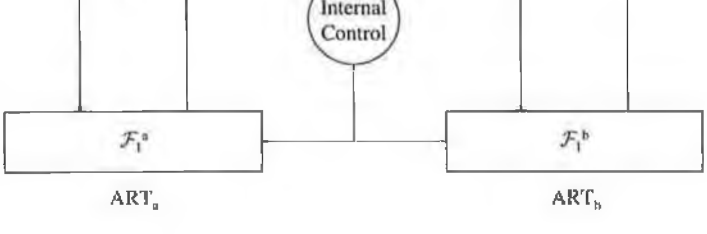


Figure 4.8: ARTMAP.

References	Feed-forward	Recurrent	Self-organizing maps	ART	ARTMAP	SONNET	Other	Application
Sano and Jenkins [140]	●							Pitch perception
Taylor and Greenhough [155]				●	●			
Scarborough et al. [141]	●							Tonality
Laden and Keefe [77]	●							
Bharucha [6], Bharucha and Todd [7]	●	●					●	
Leman [81]			●					
Griffith [59]		●	●	●				
Page [118, 119]						●		Pitch sequences and rhythm
Roberts [133]						●		
Todd [159]		●						Composition
Mozer [105, 107]		●						
Grossberg [64]				●				Stream segregation
Gjerdingen [56, 57]				●				Miscellaneous

Table 4.1: A summary of well-known applications of ANNs to music processing with respect to the types ANNs described in this chapter (see Griffith and Todd [60] and Loy [90]).

4.4.6 ARTMAP

Another variant of ART-based networks, called ARTMAP, was introduced by Carpenter et al. [21, 25] and Carpenter and Grossberg [18] to enable supervised learning to occur within the general ART framework. This architecture consists of two ART modules (ART$_a$ and ART$_b$) and an inter-ART associative memory known as a map field ($\mathcal{F}^{ab}$). ART$_a$ is fully connected to $\mathcal{F}^{ab}$ via adaptive connections, whereas ART$_b$ is connected to $\mathcal{F}^{ab}$ by fixed, bi-directional, one-to-one connections (see Figure 4.8). During training, ART$_a$ receives a stream of input vectors labelled $\boldsymbol{a}^p$ and ART$_b$ receives a stream of input vectors labelled $\boldsymbol{b}^p$. $\boldsymbol{b}^p$ represent the correct predictions associated with each vector in $\boldsymbol{a}^p$. During learning, each categorical representation at ART$_a$ is checked to see if it correctly predicts the current categorical representation at ART$_b$. If it does, these categories become associated via map field learning. If it does not, the internal control increases vigilance at ART$_a$ such that a search procedure commences. The goal of this procedure is to find a suitable category at ART$_a$ that correctly predicts the outcome at ART$_b$. If no suitable choices are found, an un-instantiated cell at ART$_a$ is chosen to learn the category and the corresponding association between ART$_a$ and ART$_b$. Thereafter, given only the set $\boldsymbol{a}^p$, the system should be able to predict the correct outcomes. For example, Carpenter et al. [21] showed how, given a set of observable features of a mushroom, ARTMAP was able to classify a mushroom as either safe or poisonous.

This model has been used extensively since its introduction; for character recognition [22, 23, 24], visual object recognition [20, 136] and medical diagnosis [19]. Within the musical domain Taylor and Greenhough [155] shows how ARTMAP was capable of classifying the pitch of four musical instruments from examples taken within the range of C3 to C6.

Table 4.1 summarizes well-known applications of ANNs to music processing with respect to the types ANNs described in this chapter. The remainder of this chapter discusses the SONNET network and the reasons behind its choice as the foundation underlying the new work presented in this thesis.

4.5 Choosing SONNET

Sections 4.4.5 and 4.4.6 briefly discussed the family of ANN's collectively referred to as Adaptive Resonance Theory, that were designed to solve the stability-plasticity dilemma. However, these networks are unable to perform context-sensitive recognition and are therefore, unable to recognize embedded patterns. Consequently, patterns must be segmented before they are presented to these networks. This poses serious questions regarding the ability of these networks to behave autonomously and in real-time in order to adequately process musical sequences. A second type of network, known as a masking field [28], has been invented to solve the problem of context-sensitive recognition. Furthermore, to ensure the stability of masking fields, Page [118] shows how to place a masking field within an adaptive resonance loop (a network known as MASKART) in a manner that solves the stability-plasticity dilemma. However, serious problems still exist with this formulation. In particular, the hard-wiring of the masking field causes a combinatorial explosion in the number of classification cells required as the number of input categories increases. However, with the conception of SONNET [112, 114], these problems have been addressed. SONNET combines the stability of ART with the capacity of masking fields to achieve context-sensitive recognition. In addition, SONNET's masking field self-organizes from an initial field of homogenous cells and connections to the heterogeneous configuration that characterizes a masking field. This overcomes the problem of the combinatorial explosion of classification cells that stifled hard-wired configurations. In fact, it is SONNET's ability to learn and recognize embedded patterns in an autonomous and robust manner that makes it an ideal choice as the core element of a CBR system (see Chapter 11). In addition, the following properties make SONNET-based networks ideal for the task of melody segmentation and retrieval:

Self-organizing using unsupervised learning. This property enables SONNET to autonomously acquire knowledge in real-time by mere exposure to a continually changing input environment. Invariance within this environment is discovered and encoded, thus enabling SONNET to create its own segmentations in response to unsegmented sequences of events.

Form stable category codes. SONNET achieves this property by freezing the LTM state of committed cell assemblies. This prevents the presentation of subsequent patterns from eroding prior learning.

Context-sensitive recognition. SONNET can learn and recognize embedded patterns in a context sensitive manner. This achieved through SONNET's self-organizing masking field and by the manner in which bottom-up links interact. One aspect of context sensitive recognition is the temporal chunking which SONNET achieves [63]. Temporal chunking allows significant categories to form, even after all of a pattern's subparts have been learned. Nigrin [114, pg. 23] cites the example of a category being formed for the word *cargo* even though categories for the words *car* and *go* have already been formed.

Extendable architecture. One major advantage of utilizing SONNET is the ease with which hierarchies can be formed. This facilitates the organization of melodies into hierarchies of phrases, thus accounting for the phrased nature of melodies. Furthermore, in Chapter 10 we will show how a parallel configuration of SONNET modules can be achieved. This enables multi-dimensional phrase hierarchies to be formed. This structure will prove invaluable during the CBR experiments presented in Chapter 13.

Generalization. Generalization is present in a variety of forms. Firstly, generalization is facilitated by varying the attention that is given (using a vigilance parameter) between an input pattern and a previously committed pattern. Consequently, SONNET is capable of performing arbitrarily coarse or fine classifications. Secondly, SONNET can vary its learning rate over a broad range, to achieve either fast, single-shot learning or slow to medium learning, which enables generalization to occur over multiple trials. Thirdly, SONNET is capable of operating within noisy input environments; by modifying its match parameter, previously learned patterns that are transformed in some way (for example, a noisy melodic query) may still be recognized. Consequently, SONNET's performance degrades gracefully as noise levels increase.

In addition to these points, SONNET has a proven track record in processing melodic sequences. In Chapter 7, research by Page [118] and Roberts [133] show how SONNET can be applied to the tasks of melody recall and rhythmic segmentation.

4.6 Summary

This chapter discussed the role and application of ANNs in music processing. Moreover, we reviewed the primary motives behind our choice of SONNET as the foundation for new research presented in this thesis. Consequently, in Part II of this thesis, we describe in greater detail the architecture and operation of SONNET, including some advanced topics and previous applications of SONNET to music processing.

Part II

SONNET Background

Chapter 5

SONNET Fundamentals

5.1 Introduction

To address the problem of performing stable classifications of temporal patterns in real-time, Nigrin [110, 111, 112, 113, 114] introduced the artificial neural network known as SONNET (**S**elf-**O**rganizing **N**eural **NET**work). In this chapter, the concepts and equations behind Nigrin's original network will be presented. Additionally, minor network modifications made by Page [118] and Roberts [133] will be discussed. This, and the remaining chapters in Part II, provide the necessary background for new SONNET-based research, which is presented in Chapters 8 through 11.

Other descriptions of Nigrin's original SONNET network can be found in Page [118, Chapter 6] and Roberts [133, Chapter 3]. However, for a more comprehensive discussion of this material, see Nigrin [112, 114].

5.2 Overview of SONNET

5.2.1 Properties of SONNET

SONNET is capable of responding, in real-time, to a continuous stream of *real-world* input events. That is, events are unlabelled and noise is ubiquitous. Within such an environment, SONNET's performance degrades gracefully as noise levels increase. With ongoing exposure to such an environment, SONNET can self-organize, using unsupervised learning, to form stable category codes that represent meaningful patterns which are present within this environment. SONNET can vary its learning rate over a broad range, to achieve either fast, single-shot learning or slow to medium learning that enables generalization to occur over multiple trials. Generalization may also be facilitated by varying the care that is given (using a vigilance parameter) between an input pattern and a previously learned archetype. This enables SONNET to perform arbitrarily coarse or fine classifications.

On account of its masking field structure, SONNET can classify embedded patterns in a context-sensitive manner. In other words, SONNET can classify patterns that are surrounded by extraneous

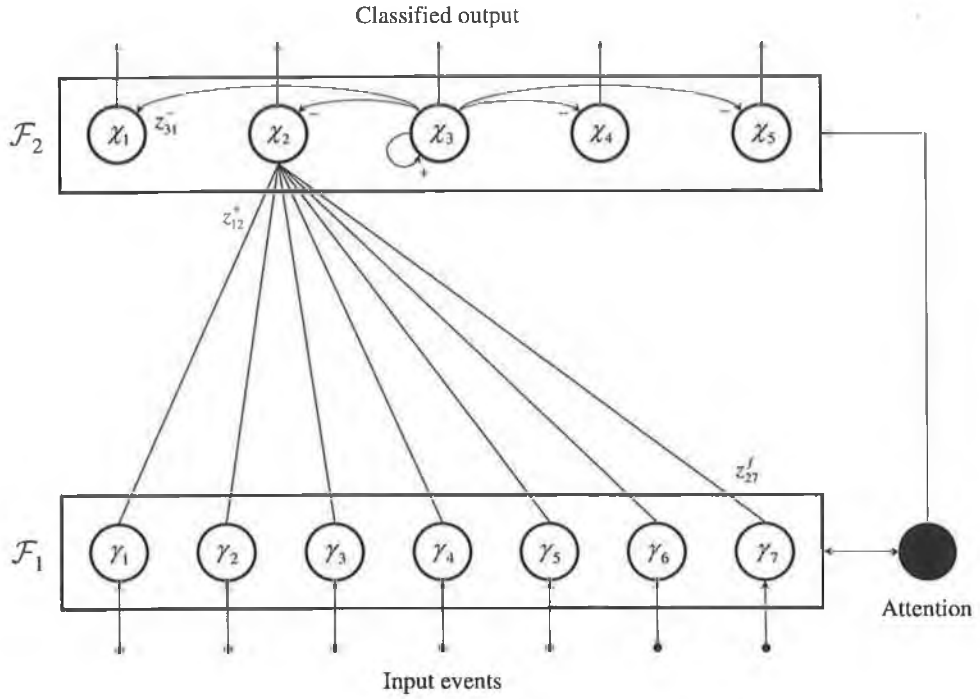


Figure 5.1: The SONNET architecture. Note that each γ_j-χ_i pair are actually connected by separate unidirectional bottom-up and top-down links (as shown in Figure 5.2). Here, however, for simplicity, these links are combined and shown as bi-directional connections.

information. However, if this additional information is not extraneous and instead contributes to form a larger pattern, then SONNET can learn this larger pattern by *masking-out* smaller, previously learned sub-patterns. In this way, SONNET creates its own segmentations in response to input sequences; thus patterns do not have to be pre-segmented before they are classified. This is a fundamental property of any autonomous system.

5.2.2 Architecture and operation of SONNET

A SONNET module consists of two highly interconnected fields of cells, $\mathcal{F}_1$ and $\mathcal{F}_2$ (see Figure 5.1). $\mathcal{F}_1$ is configured as a working memory that stores the most recently presented input events and the order in which they occurred, thereby transforming a sequential input pattern into a spatial activity pattern. $\mathcal{F}_1$ contains N_1 input assemblies[1] (γ_j), each of which responds to a distinct type of event. γ_j consists of an *input cell* (s_j) and a *feedback cell* (f_j). $\mathcal{F}_2$ contains N_2 cell assemblies (χ_i), each of which attempts to classify patterns that occur across $\mathcal{F}_1$. χ_i contains an *excitatory cell* (b_i), a *classification cell* (c_i), a *feedback cell* (d_i) and an *inhibitory cell* (e_i) (see Figure 5.2).

Each γ_j will activate when its associated event type occurs. These activation signals are propagated to χ_i via an excitatory weight z^+_{ji} that is incidental to each bottom-up link. There, χ_i

[1] The term γ_j, denoting an input assembly, is introduced in this thesis to acknowledge the use of more than one cell in processing each event occurrence at $\mathcal{F}_1$. This is consistent with the labelling scheme used at $\mathcal{F}_2$, which uses χ_i to label each cell assembly.

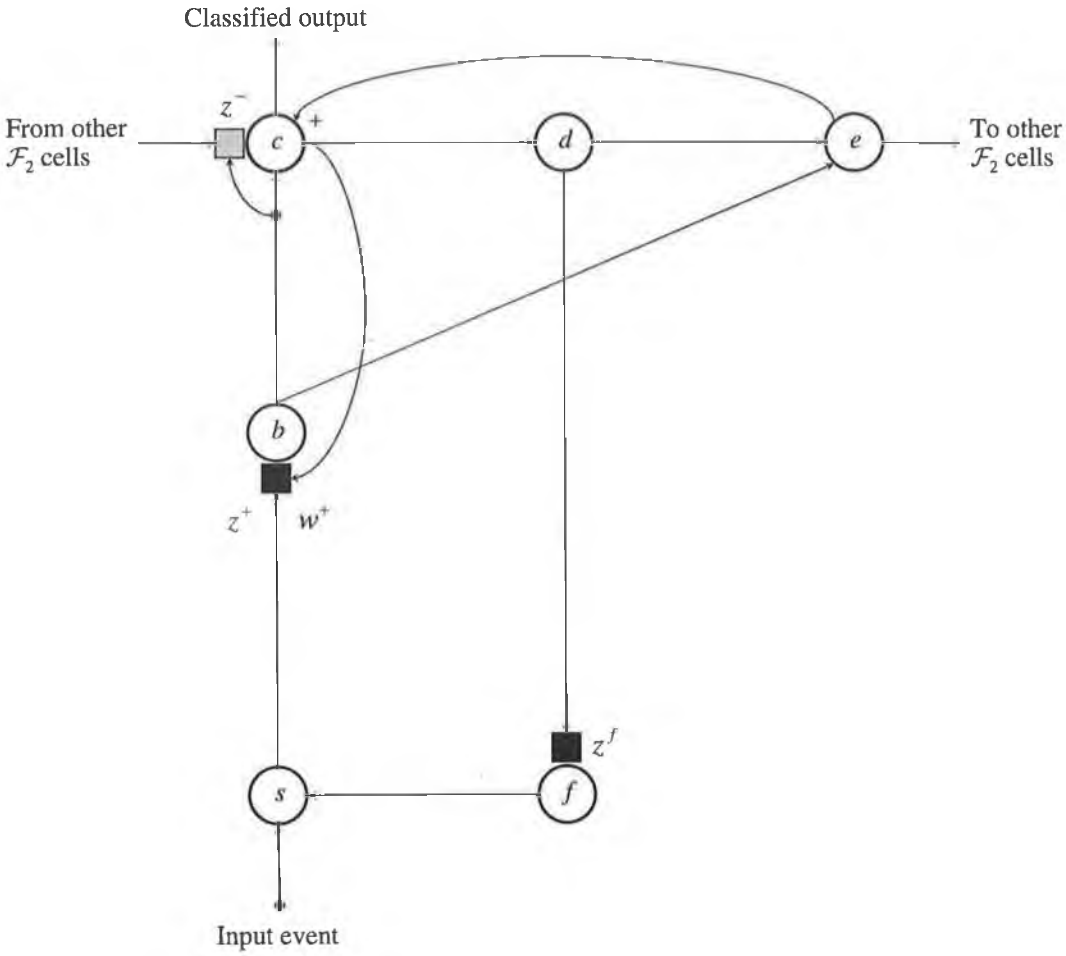

Classified output
From other $\mathcal{F}_2$ cells
To other $\mathcal{F}_2$ cells
z^-
+
c
d
e
b
z^+
w^+
z^f
s
f
Input event

Figure 5.2: The cells, links and weights that define a γ_j-χ_i pair.

competes (via the inhibitory weights z_{ji}^-) with other cell assemblies at $\mathcal{F}_2$ to learn and classify the spatial pattern across $\mathcal{F}_1$. χ_i learns by adjusting its excitatory weights to become parallel to this activity pattern. When χ_i has fully learned to classify a pattern it freezes its excitatory weights and is said to be committed ($\chi_i^{com} = true$), thus forming a stable category code. The input assemblies that form χ_i's pattern become members of χ_i's receptive field, which is specified by the set T_i. In turn, T_i is derived from the secondary weights w_{ji}^+. Thereafter, when χ_i recognizes its pattern at $\mathcal{F}_1$, the input assemblies that form the set T_i will be reset (using the feedback links from $\mathcal{F}_2$ to $\mathcal{F}_1$). This process, known as *chunking-out*, helps to allow sequences that are greater than the *transient memory span* (TMS) to be processed (see Section 5.3.1).

As learning progresses, the $\mathcal{F}_2$ field self-organizes from an initially homogeneous layer into a non-uniform layer called a masking field. Each cell assembly is said to have a size (denoted by D_i) that is dependent on $|T_i|$. Therefore, D_i increases with the length of the sequence it classifies. This enables larger cell assemblies to overpower smaller cell assemblies, which leads to unequal competition between cell assemblies of different sizes. It is this unequal competition that facilitates context sensitive recognition and enables the *temporal chunking problem* to be solved (see Section 5.3.6). Furthermore, the inhibitory weights (z_{ji}^-) within a masking field evolve such that only cell assemblies that classify similar patterns will compete with one another, where similarity is defined in terms of the vigilance parameter ρ. Consequently, cell assemblies that do not classify similar patterns will not compete and, instead, may co-activate.

The top-down feedback weights (z_{ij}^{f}) are modified in an analogous way to the excitatory bottom-up weights; thus the two sets of weights become symmetrical. The feedback weights aid with the chunking-out process and may be utilized to provide plausible expectancies based on the current input; in other words, recall previously learned patterns (see Section 7.2).

5.3 Cell and weight dynamics

This section describes in detail how the cells and weights of the SONNET network evolve to behave in the manner described thus far.

5.3.1 Configuring $\mathcal{F}_1$ as a working memory

This section describes how a continuous sequence of arbitrary temporal events can be transformed into a non-ambiguous spatial pattern of activity across $\mathcal{F}_1$. Thus, $\mathcal{F}_1$ acts as a working memory that temporarily stores event occurrences whilst preserving the order in which they occurred.

Storing order information using a decreasing activity pattern

Each $\mathcal{F}_1$ input assembly represents a distinct type of event.[2] When such an event occurs, the corresponding input assembly (γ_j) activates (or fires) by setting its input cell (s_j) to an initial value of μ. Subsequently, this activation level begins to increase at a constant rate ω. In this way, cells that fired in response to early events have higher activations than those that occurred thereafter. Thus, order information is stored by a decreasing pattern of activity across $\mathcal{F}_1$. Patterns stored in this way may also be referred to as *monotonically decreasing* or being stored with a *primacy gradient* (see Figure 5.3).

The excitatory bottom-up and top-down weights, within a cell assembly's receptive field (specified by the set T_i), evolve to become parallel to the normalized activity pattern across $\mathcal{F}_1$ (see Sections 5.3.3 and 5.3.7). Consequently, patterns in LTM are also stored with a primacy gradient, mirroring its STM representation at $\mathcal{F}_1$. This enables the top-down feedback weights (z_{ij}^{f}) to generate plausible expectations in response to the current input at $\mathcal{F}_1$ (see Section 7.2 for more detail). In contrast, if increasing activity patterns are employed (where the activation levels of input cells decay with time), the network would generate expectancy signals that are biased towards the events at the end of a learned pattern rather than towards events occurring in the correct sequential order with respect to the pattern [114, Section 2.2.1] (increasing activity patterns may

[2] An event does not have to be a musical event such as a note onset or a drum beat. Events may include letter occurrences or spoken syllables for example. See Section 14.3.4 for a more detailed discussion on other possible applications of SONNET.

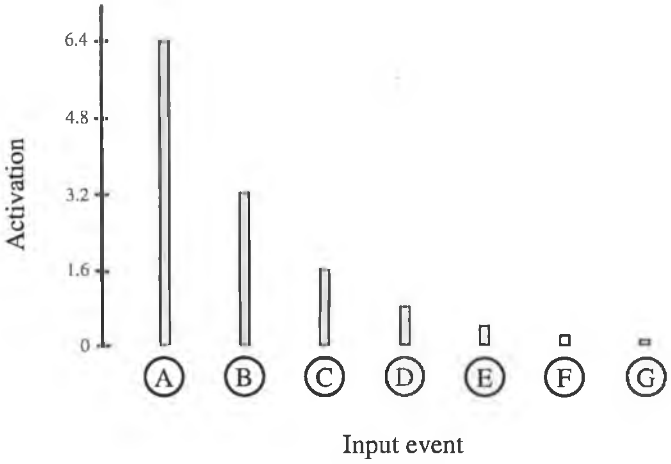


Figure 5.3: The activity trace of $\mathcal{F}_1$ in response to the sequence {A, B, C, D, E, F, G}. This sequence could be considered as consisting of letter occurrences or notes from the *tonic sol-fa*. In this example $\mu \approx 0.1$ and $\omega \approx 2$.

also be referred to as *monotonically increasing* or as being stored with a *recency gradient*). Although neural networks exist that can produce expectancies using increasing storage schemes, the circuitry of these networks are more complicated.

Another disadvantage of using recency gradient storage schemes with SONNET is that they can lead to circumstances whereby a cell assembly responding to the beginning of its pattern could receive less bottom-up input than a cell assembly that is responding to the end of its pattern. For example, suppose χ_{AB} and χ_{BA} have learned the sequences {A, B} and {B, A} respectively. Furthermore, suppose the item A has just been presented at $\mathcal{F}_1$. The use of a recency gradient would cause χ_{BA} to receive more gated bottom-up input than χ_{AB}, which is inconsistent with that fact that the start of χ_{AB}'s pattern is present at $\mathcal{F}_1$ [133, Section 3.2.3]. This problem does not occur when using primacy gradient storage schemes.

Restrictions on μ and ω

To achieve a consistent primacy gradient across $\mathcal{F}_1$, thus allowing $\mathcal{F}_2$ to code input sequence regularities and indeed recognize previously learned patterns, a number of restrictions need to be placed on the values of μ and ω. Firstly, to ensure that s_j is always non-negative, μ must be greater than zero ($\mu > 0$). Secondly, to achieve a primacy storage gradient, the shunting parameter ω must be greater than unity ($\omega > 1$). If ω was allowed to be less than unity (but greater than zero), patterns would be stored with a recency gradient which, as we have already discussed, is undesirable.

Furthermore, both μ and ω should remain constant. This facilitates the learning of input sequence regularities by cell assemblies at $\mathcal{F}_2$. This is because, with constant parameters, the continuing transformation of temporal events into a spatial pattern of activity across $\mathcal{F}_1$ will preserve the ratio between input cells that activated in response to previous input events. This solves a special case of the *LTM invariance principle* that is relevant to SONNET [114, Section 2.2.2].

Implementing $\mathcal{F}_1$ as an on-centre off-surround network

The short-term storage of temporal events as a decreasing pattern of activity across $\mathcal{F}_1$ can be implemented using an on-centre off-surround network. The number of connections required to implement such a neural circuit is of the order $O(n^2)$, where n is the number input assemblies in $\mathcal{F}_1$ ($n = N_1$). However, the number of connections can be reduced to $O(n)$ by employing the use of a global cell, g_s, which calculates total field activity. The following input cell activation equation can implement the behaviour described so far:

$$\frac{\partial}{\partial t} s_j = (B_s - s_j)[v_s s_j + I_s] - v_s s_j [g_s - s_j] \qquad (5.1)$$

$$g_s = \sum_{j \in \mathcal{F}_1} s_j \qquad (5.2)$$

where B_s is a constant that represents the maximum activity level that s_j may reach, v_s is a fixed connection weight on the link from g_s to each input cell and I_s is the external input applied to s_j. I_s is only non-zero for a time period of K_{16} after an event has occurred at s_j.

Nigrin [112, 114] shows how this circuit can achieve values of $\mu \approx 0.1$ and $\omega \approx 2$ for an isochronously presented input pattern by using the following parameter values: $B_s = 50$, $v_s = 0.075$, $I_s = 0.007$ and $K_{16} = 0.4$ seconds. Similarly, Roberts [133] shows how values of $\mu \approx 0.003$ and $\omega \approx 3$ can be achieved over the period of a tactus-span by using the following values: $B_s = 100$, $v_s = 0.0187$, $I_s = 0.0005$ and $K_{16} = 0.06$ seconds.

In summary, when an input event occurs, the appropriate input cell is set to μ. Thereafter, the cell increases at a constant rate ω. However, constant values for μ and ω cannot be maintained for arbitrarily long lists using Equation 5.1. Therefore, an input cell reset mechanism is required in order to limit the depth and capacity of working memory at $\mathcal{F}_1$. The manner by which this can be achieved is discussed next.

Limiting the depth and capacity of $\mathcal{F}_1$

Retaining order information becomes increasingly difficult, using Equation 5.1, because the total field activity will eventually saturate (or overload). The maximum number of items that can be stored before order information becomes confused is called the TMS (note that this type of confusion is consistent with studies of human STM memory [114, pg. 78]). By altering the parameters B_s, v_s, I_s and K_{16}, the TMS may be varied. Nonetheless, to process sequences that are longer than the TMS, it is necessary to restrict the depth and capacity of short-term storage at $\mathcal{F}_1$.

When cell assemblies at $\mathcal{F}_2$ have learned to classify input patterns, a process known as *chunking-out* (or *rehearsal*) is usually sufficient to prevent $\mathcal{F}_1$ overload (see Section 5.3.8). However, before then, another reset process is required. Generally, this process monitors the total field activity of $\mathcal{F}_1$ (remember, total field activity is represented by g_s). Then, when g_s reaches or exceeds a

Event	Active input cells	g_s
e_1	A	0.1
e_2	AB	0.3
e_3	ABC	0.7
e_4	ABCD	1.5
e_5	ABCDE	3.1
e_6	ABCDEF	6.3
e_7	DEFG	1.5
e_8	DEFGH	3.1
e_9	DEFGHI	6.3
e_{10}	HIJ	0.7

Table 5.1: $\mathcal{F}_1$ overload with $\mu \approx 0.1$, $\omega \approx 2$, $K_{12} = 10$ and $K_{11} = 1$. Just after the seventh event occurs, g_s will become greater than K_{12}. In our current example this causes the three most active input items (A, B and C) to be reset (due to K_{11}). In this case, the next most highly active input cell (representing D) was not reset (remember, there is a 50% chance of this happening). Subsequently, just after the tenth event, g_s again becomes greater than K_{12}. In this case, due to K_{11} and the random reset element, the four most active input cells are reset (D, E, F and G).

predetermined threshold of K_{12}, the field is considered to have overloaded. In this circumstance, certain input cells are reset in order to reduce the total field activity to below K_{12}.

One possibility is to reset every active input cell at $\mathcal{F}_1$ when $g_s \geq K_{12}$. However, Nigrin [114, Section 4.5] dismisses this method because it is highly probable that patterns occurring at $\mathcal{F}_1$, which have already been learned at $\mathcal{F}_2$, will be deleted (or partially deleted) from $\mathcal{F}_1$ before they can be recognized. For example, suppose a maximum of seven input items can be represented at $\mathcal{F}_1$ before the field overloads. Furthermore, suppose χ_{DEFG} has learned the pattern {D, E, F, G} and the input sequence {A, B, C, D, E, F} has just been presented. Now, the presentation of one further item, for example G, would cause every input item to be removed from $\mathcal{F}_1$ (including G). Consequently, χ_{DEFG}, will not have had a chance to recognize and classify its pattern because it has been deleted from $\mathcal{F}_1$.

Another possibility is to reset the most highly active input cell when $g_s \geq K_{12}$. This *sliding window* approach would be unsuitable for an event-driven network (see Section 6.3) because cell assemblies would not have had a large enough response time to process any given pattern before its first item is deleted from $\mathcal{F}_1$ (apart from patterns at the beginning of a sequence). However, if one wishes to let time have a greater influence on processing, then this scheme would be suitable, particularly for limiting the depth of STM [133, pg. 132]. This approach is discussed further in Section 7.3.2.

A final possibility is to reset all input assemblies whose activity level exceed a specified threshold of K_{11} when $g_s \geq K_{12}$. This threshold can be varied to enable a particular number of input cells to be removed from $\mathcal{F}_1$. Additionally, Nigrin [114, Section 4.5] suggests the use of a random element in this reset mechanism. Indeed, Nigrin and Roberts have successfully used variations of this approach to limit STM capacity. For example, Roberts [133] suggests that $g_s \geq K_{12}$ should occur after seven items in order to model the capacity of human STM. Furthermore, upon overload, the three most highly activated input cells should be deleted from working memory (specified using K_{11}). A fourth item should then be deleted with a 50% probability. This scenario assumes four input assemblies reach the K_{11} threshold (for example, see Table 5.1). The use of a random component ensures that resets do not occur in the same place, although as Page [118, pg. 141] points out with respect to his SONNET experiments, *'the random element in the reset process often resulted in a working memory pattern representing an "across phrase" group of notes'*. This problem is difficult to overcome without the use of other grouping cues to help with discovering boundary points between phrases (such as Roberts' method of using time in the segmentation process, see Section 7.3).

5.3.2 Excitatory bottom-up input to $\mathcal{F}_2$

Input cell activations at $\mathcal{F}_1$ are sent via the bottom-up connections to $\mathcal{F}_2$. These signals are are gated by the excitatory weights (z^+_{ji}) before reception at each excitatory cell b_i (see Figure 5.2). Thus, the activation level of b_i represents χ_i's excitatory gated input. This excitatory signal is then forwarded to each classification cell c_i and each inhibitory cell e_i (see Section 5.3.5). The gated input is not calculated using the usual dot product (that is, $\sum\limits_{j=1}^{N_1} s_j z^+_{ji}$), because this method would not allow small weights to significantly affect total input to b_i. This is due to the fact that the excitatory weights converge to decreasing activity patterns (see Section 5.3.3). Instead, the following non-linear rule is employed:

$$b_i = I^+_i I^\times_i \tag{5.3}$$

$$I^+_i = \sum_{j=1}^{N_1} S_{ji} z^+_{ji} \tag{5.4}$$

$$I^\times_i = \max\left(\prod_{j \in T_i} I^\times_{ji}, K_8\right) \tag{5.5}$$

$$I^\times_{ji} = K_1 + K_2 \min\left(1, \frac{S_{ji}}{Z_{ji}}\right) \tag{5.6}$$

$$Z_{ji} = \frac{z^+_{ji}}{\left[\sum_{j \in T_i} (z^+_{ki})^2\right]^{1/2}} \tag{5.7}$$

$$S_{ji} = \begin{cases} \dfrac{s_j}{\left[\sum_{k \in T_i} (s_k)^2\right]^{1/2}} & \text{if } j \in T_i \\ \dfrac{s_j}{\left[\sum_{k=1}^{N_1} (s_k)^2\right]^{1/2}} & \text{otherwise} \end{cases} \tag{5.8}$$

I_i^+ represents the dot product between the normalized input vector at χ_i and the excitatory weight vector. I_i^+ has a maximum possible value of 1.0, which can only be achieved by committed cell assemblies that respond to their full pattern. The normalization of inputs at χ_i (represented by S_{ji}) is dependent on membership of χ_i's receptive field, which is specified by the set T_i (see Section 5.3.4). This dependence has the advantage of allowing patterns to be classified even when they are embedded within arbitrarily large input sequences. Furthermore, before a cell assembly has fully encoded a pattern, the S_{ji} calculation allows input assemblies that are not part of T_i to influence processing and perhaps over time to become members of T_i.

The value $I_i^\times$ compares the normalized weight vector against the normalized input vector, thus measuring the confidence with which a cell assembly matches the input sequence at $\mathcal{F}_1$. The constants K_1 and K_2 should be initialized such that: $0 < K_1 < 1$ and $K_1 + K_2 > 1$. Together, these constants strongly influence the values that $I_i^\times$ may reach. For example, the maximum value $I_i^\times$ may reach with $|T_i| = 4$ and values of $K_1 = 0.5$ and $K_2 = 1.5$ is $(K_1 + K_2)^{|T_i|} = 16$. Furthermore, in this example, the absence of the final item in T_i would give a value of $I_i^\times \leq 4$. Therefore, the presence or absence of input items encoded by small weights significantly affect total input to χ_i. Finally, the constant K_8 limits the minimum value that $I_i^\times$ may reach. This restriction prevents excessively slow learning times; however, it may also prevent χ_i from distinguishing between different levels of mismatch. More details on the design of the confidence formulation are discussed in Section 8.3.

5.3.3 Learning excitatory bottom-up weights

The categories formed at each cell assembly are encoded through the excitatory weights (z_{ji}^+). These excitatory weights are considered as part of SONNET's LTM and enable a cell assembly to recognize the STM pattern at $\mathcal{F}_1$ from which these weights originally converged. Each distinct category that is learned will be encoded by a single cell assembly at $\mathcal{F}_2$. A category at χ_i is considered to have formed when the gated input, represented by I_i^+, reaches unity. This occurs when the LTM weights become parallel to a portion of the decreasing activity pattern across $\mathcal{F}_1$. Thereafter, a cell assembly is committed to this pattern and χ_i^{com} is defined as true. In this circumstance the excitatory weights are frozen; thus a stable category code is formed. However, before a category is stabilized, the excitatory weights evolve (from initial values specified randomly within the range $[K_{zmin}, K_{zmax}]$) according to the following differential equation:

$$\frac{\partial}{\partial t} z_{ji}^+ = \epsilon_1 r_{ji} c_i \left[-L_i z_{ji}^+ + S_{ji} c_i \right] \tag{5.9}$$

$$L_i = \begin{cases} 1 & \text{if } z_i^{tot} > z_{min} \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

$$z_i^{tot} = \left[\sum_{k \in T_i} (z_{ki}^+)^2 \right]^{\frac{1}{2}} \tag{5.11}$$

ϵ_1 represents the learning rate and L_i is a LTM decay function. L_i prevents the LTM weights from becoming too small, with z_{min} specifying the lower limit. This prevents a dramatic reduction in learning times under slow learning conditions. z_i^{tot} represents the length of the LTM weight vector.

The classification cell, c_i, contributes to the rate of change of the LTM weights, with larger activities causing faster learning. c_i also shunts the term S_{ji}; this allows learning to take place at uncommitted cell assemblies without compelling those cell assemblies to fully encode a pattern. This is a vital property in forming stable category codes because even small c_i activities enable weights to become parallel to the activity traces at $\mathcal{F}_1$. Note that adjustments to z_{ji}^{+} are based on the activity levels of c_i even though the two values are disassociated from one another. This separation makes possible the correct operation of inhibitory learning (see Section 5.3.6).

One final value, crucial to the correct behaviour of the excitatory learning equation, is the *learning-rate modulator*, r_{ji}. This quantity was introduced by Nigrin [114, Section 3.14.4] to prevent cell assemblies from averaging different input patterns and forming an archetype that was never part of the input sequence in the first place. In other words, r_{ji} prevents a cell assembly, χ_i, from learning more than one pattern. This problem can materialize in two different ways. Firstly, when different patterns consist of the same input items. For example, if the patterns {A, B} and {B, A} are alternately presented, it is just as likely that a cell assembly will learn an average of the two patterns as opposed to either of the individual patterns (see Experiment 7 in [114, Section 3.15]). A second manifestation of the problem occurs when an cell assembly is attempting to learn a pattern that is embedded in different contexts. For example, to generalize from the patterns {C, A, B}, {D, A, B} and {E, A, B} and learn the pattern {A, B}. Without the learning-rate modulator, significant weights from s_C, s_D and s_E will be learned (see Experiment 8 in [114, Section 3.15]). The learning-rate modulator is calculated as follows:

$$r_{ji} = \begin{cases} 1.0 & \text{if } z_i^{tot} < K_5 + K_6 \\ \left(\frac{z_i^{tot} - K_5}{K_6}\right)^{1-M_i^{-4}} & \text{if } j \in T_i \\ \left(\frac{z_i^{tot} - K_5}{K_6}\right)^{1-(1-M_i)^{-4}} & \text{otherwise} \end{cases} \tag{5.12}$$

$$M_i = \min\left(1, \frac{I_i^{\times}}{I_i^{max}}\right) \tag{5.13}$$

$$\frac{\partial}{\partial t} I_i^{max} = \begin{cases} 0 & \text{if } \chi_i^{com} = \text{true} \\ \epsilon_1 \left[-I_i^{max} + 2I_i^{\times}\right] & (\chi_i^{com} = \text{false}) \text{ and } (I_i^{max} < I_i^{\times}) \\ -\epsilon_1 r_{0i} c_i I_i^{max} & (\chi_i^{com} = \text{false}) \text{ and } (I_i^{max} \geq I_i^{\times}) \end{cases} \tag{5.14}$$

K_5 is set to the z_i^{tot} value that χ_i obtains when it is judged to have a fair pattern representation. Typically, K_5 is set to between 0.25 and 0.3. K_6 regulates the magnitude of the weight changes as z_i^{tot} increases above K_5. K_6 is typically set to 0.001. M_i measures how well χ_i represents the current pattern by comparing $I_i^{\times}$ with the maximum confidence that can be expected (I_i^{max}) at χ_i. $M_i \approx 1.0$ indicates a good match, while $M_i \approx 0.0$ indicates a poor match. Finally, r_{0i} is the

r_{ji} value used to modulate excitatory weights that are members of T_i. That is,

$$r_{0i} = \left(\frac{z_i^{tot} - K_5}{K_6}\right)^{1-M_i^{-4}} \tag{5.15}$$

5.3.4 Specifying the set T_i

A cell assembly's receptive field is specified by the set T_i, which consists of the set of indices that identify the input cells at $\mathcal{F}_1$ that are considered to form χ_i's pattern. Initially, every connection from s_j is considered to be part of T_i. In other words, before learning has taken place, $\mathcal{F}_2$ is fully connected to $\mathcal{F}_1$. However, as learning progresses, χ_i's receptive field is effectively pruned, allowing only significant inputs to influence processing at χ_i. Biological evidence supports this type of synapse reduction as part of the learning process. For example, Squire and Kandel [154, pg. 146] describe the process of LTM as being *"associated with the growth of new synaptic connections or the retraction of pre-existing ones"*.

An excitatory weight (z_{ji}^+) cannot be used to determine membership of the set T_i because it is ambiguous from the magnitude of z_{ji}^+ whether or not an input item has been presented frequently enough to become a member [114, pg. 143]. Therefore, a secondary weight, w_{ji}^+, is employed at the end of each connection from s_j to χ_i. w_{ji}^+'s magnitude will determine whether s_j should be included in the set T_i by measuring how often s_j is active while a particular classification cell (c_i) is simultaneously active. Initially, each w_{ji}^+ is initialized to the same value specified by K_{winit} (for example, 0.02). Thereafter, before the secondary weights at χ_i become frozen, in the same way as the excitatory weights at χ_i do, they evolve in the following way:

$$\frac{\partial}{\partial t} w_{ji}^+ = \begin{cases} \epsilon_1 r_{ji} c_i \left[-L_i w_{ji} + c_i\right] & \text{if } K_4 S_{ji} > S_{0i} \\ \epsilon_1 r_{ji} c_i \left[-L_i w_{ji}\right] & \text{otherwise} \end{cases} \tag{5.16}$$

where ϵ_1, r_{ji} and L_i are the same values used in the excitatory weight calculations. S_{0i} is the largest normalized input at χ_i. K_4 is a constant greater than unity which specifies which inputs are significant and which inputs are treated as noise. Since input cells with activation levels below $\frac{S_{0i}}{K_4}$ are treated as noise, K_4 determines the maximum pattern size that may be learned by a cell assembly. For example, setting $K_4 = 12$ with input parameters of $\mu \approx 0.1$ and $\omega \approx 2$ will restrict $\mathcal{F}_2$ cell assemblies to learning patterns with a maximum length of four items.

In contrast to the excitatory weight learning equation, secondary weights from active input cells to χ_i will increase or decrease by an equal amount regardless of the activity level of any particular input cell. Consequently, the secondary weight on the link from s_j to χ_i measures how often s_j contributes to the activation of χ_i relative to other secondary weights on other links to χ_i. As a result, for s_j to be considered as a member of the set T_i, w_{ji}^+ on that link must be relatively large. More specifically, the following condition must be satisfied:

$$j \in T_i \iff K_3 w_{ji}^+ \geq w_{0i}^+ \tag{5.17}$$

where w_{0i}^+ is the largest secondary weight to χ_i and K_3 is a parameter (greater than unity) used to control how difficult it is for a link to be considered as a member of T_i, with larger values making membership easier. In this respect K_3 can be considered a vigilance parameter that controls generalization at the link level. Typically, K_3 is set between 1.3 and 2.0.

5.3.5 χ_i dynamics

Calculating activation levels at χ_i

The activation level of χ_i is an indication of how well it represents the activity pattern across $\mathcal{F}_1$ with respect to other cell assemblies. χ_i's activation level is given by the classification cell c_i, which evolves as follows:

$$\frac{\partial}{\partial t}c_i = -Ac_i + \left(\frac{B - c_i}{D_i}\right)(v_1 b_i + e_i) - \frac{c_i}{D_i} v_2 Q_i \tag{5.18}$$

where A represents a passive decay constant, B is the maximum activation level c_i may reach, v_1 and v_2 are fixed weights that can be used to vary the ratio between excitatory and inhibitory input to c_i. D_i represents the dilution parameter or cell size, which will be described further in Section 5.3.6. b_i represents the gated excitatory bottom-up input and has already been discussed (see Section 5.3.2). The gated inhibitory input (Q_i) to a cell assembly is given by:

$$Q_i = \sum_{j \neq i} e_j z_{ji}^{-} \tag{5.19}$$

A ceiling of $(e_i^{big} K_7)$ is placed on Q_i for uncommitted cell assemblies, where e_i^{big} represents the largest e_j from any uncommitted cell assembly (including e_i) to c_i. K_7 regulates the total inhibition that can be received by uncommitted cell assemblies. This ceiling prevents the network from taking an excessive amount of time to learn new patterns that are similar to patterns that have already been learned.

The inhibitory cell e_i is used to facilitate competition between cell assemblies that are competing to classify their respective patterns across $\mathcal{F}_1$. e_i sends an excitatory signal to c_i (on-centre) and inhibitory signals to all the other classification cells c_j in $\mathcal{F}_2$ (off-surround). Its activation level is computed as follows:

$$e_i = b_i d_i^2 \tag{5.20}$$

The feedback cell d_i is used to send feedback signals to $\mathcal{F}_1$. d_i's maximum value is restricted to 1.0 in order to place a ceiling on the inhibitory signals by which cell assemblies compete. Feedback signals are used during the chunking-out process to reset the classified portion of an input pattern at $\mathcal{F}_1$, which is specified by T_i (see Section 5.3.8 for more details). d_i's activation level is computed as follows:

$$d_i = \min(c_i, 1.0) \tag{5.21}$$

Note that e_i and d_i are required only to enable feedback signals to be sent properly down a network hierarchy (see Section 7.2).

Classification cell reset mechanisms

A classification cell, c_i, is reset to zero if the following conditions are simultaneously met:

- There exist at least q uncommitted assemblies, χ_j, such that $K_{10}e_j > e_i$
- There exist at least q uncommitted assemblies, χ_k, such that $I_k^\times > I_i^\times$

The q number of uncommitted cell assemblies in each case need not be the same. These conditions restrict the number of uncommitted cell assemblies that are allowed to activate in response to input patterns at $\mathcal{F}_1$. This prevents too many uncommitted cells from partially encoding the same pattern. Consequently, a disjoint pattern that is presented relatively infrequently to other patterns can be learned [114, Section 3.14.3]. The constant K_{10} (which must be less than or equal to 1.0) specifies how inadequately a cell assembly must represent a pattern before it is shut-off.

An additional benefit of this c_i reset mechanism involves a reduction in the number of cell assemblies that are adjusting their LTM weights at any one point in time (provided q is set to relatively small values). This helps to reduce computation time because shut-off cell assemblies need not adjust their activations or their weights.

5.3.6 Self-organizing $\mathcal{F}_2$ into a masking field

Masking fields are characterized by the heterogeneity of both their cells and the pattern of the connections between those cells. Cell assemblies are heterogeneous in that they are considered to have a size that is dependent on the length of the pattern they code. Inhibitory connections are heterogeneous in that connections only exists between cell assemblies that represent similar patterns. This non-uniformity, coupled with an inherent bias towards the formation of large patterns over small patterns, enables masking fields to solve the temporal chunking problem and perform context sensitive recognition. Furthermore, the formation of larger patterns leads to better sequence compression and larger patterns also have greater predictive power than smaller patterns. This section describes how an initially uniform field of cell assemblies and connections can self-organize into a masking field at $\mathcal{F}_2$.

Determining the size of a cell assembly

$\mathcal{F}_2$ consists of N_2 heterogeneous cell assemblies. This heterogeneity stems from the fact that each cell assembly, χ_i, is characterized by its size, D_i, which is derived from $|T_i|$. Large D_i values imply the encoding of large patterns. Initially, D_i is identical at every cell assembly (D_i is initialized to unity); however, over the course of learning, D_i is refined to reflect changes in $|T_i|$. Consequently, D_i is considered to be a self-organizing quantity.

D_i is also known as a dilution parameter because in dilutes both the excitatory input and inhibitory input (see Equation 5.18). This means that larger cell assemblies are more difficult to turn on and off. However, once they are turned on they can easily suppress smaller cell assemblies because they output more inhibition. Therefore, with this unequal competition, a cell assembly's activity can be suppressed by a larger cell assembly even when it receives maximum input. Such a cell is said to be *masked-out*. It is this masking-out that solves the temporal chunking problem and enables context sensitive recognition.

When χ_i becomes committed, D_i is frozen in the same way excitatory weights are. However, before then, D_i evolves in the following way:

$$\frac{\partial}{\partial t} D_i = \epsilon_1 \epsilon_2 c_i \left[-D_i + I_i^{\times} \right] \tag{5.22}$$

In effect, D_i tracks $I_i^{\times}$, which in turn is derived from $|T_i|$.

Learning inhibitory weights

This section shows how the inhibitory weights (z_{ji}^{-}) can be modified to allow a uniform inhibitory connectivity pattern to evolve into a non-uniform inhibitory connectivity pattern, which is characteristic of a masking field. A pair of cell assemblies should only compete if they classify similar patterns, in which case the inhibitory weights between them should equilibrate to 1. If a pair of cell assemblies (χ_j and χ_i) represent patterns that are not similar, the inhibitory weights between them should equilibrate to 0. Consequently, if the patterns that χ_j and χ_i encode do not overlap, that is $T_j \cap T_i = \emptyset$, they may be classified simultaneously. Note that values of 1 and 0 do not accurately reflect the level of similarity between the two patterns. In Chapter 7 an inhibitory learning equation, introduced by Roberts, will describe one method of more accurately reflecting this degree of similarity.

The inhibitory weight from χ_j to χ_i is modified according to the following equation:

$$\frac{\partial}{\partial t} z_{ji}^{-} = \begin{cases} \epsilon_1 \epsilon_2 c_i \left[-z_{ji}^{-} + 1 \right] & \text{if } (\rho_i \leq I_j^{\times}) \text{ or } (\chi_i^{com} = \chi_j^{com} = \text{true}) \\ \epsilon_1 \epsilon_2 c_i \left[-z_{ji}^{-} - 1 \right] & \text{if } (\rho_i > I_j^{\times}) \text{ and } (\chi_i^{com} = \chi_j^{com} = \text{false}) \text{ and } (z_{ji}^{-} > 0) \\ 0 & \text{otherwise} \end{cases} \tag{5.23}$$

where ρ_i is a vigilance parameter used to control the coarseness of the categories to be formed. High vigilance parameters will cause new patterns, which differ only slightly from previously learned patterns, to be treated as a novel. Attempts will be made to learn these patterns. Conversely, low vigilance setting will allow patterns to be recognized as familiar even if relatively significant differences exist between these and previously learned patterns. No attempt will be made to learn these patterns. In other words, ρ_i is used to specify the level of generalization that may occur. ρ_i is calculated as follows:

$$\rho_i = \begin{cases} (I_i^{\times})^{\rho} & \text{if } I_i^{\times} \geq 1 \\ (I_i^{\times})^{1/\rho} & \text{otherwise} \end{cases} \tag{5.24}$$

An inhibitory weight freezing mechanism is utilized to enable cell assemblies that encode non-overlapping (or disjoint) patterns to co-activate. This is achieved by setting and freezing the inhibitory weights between these cell assemblies to zero. For example, suppose the disjoint patterns {A, B} and {C, D} have been learned by χ_j and χ_i respectively. In this case, it is desirable that χ_j and χ_i do not compete. To achieve this behaviour, learning between committed cell assemblies occurs differently. The weights between a pair of committed cell assemblies χ_j and χ_i are set to and frozen at zero if the following conditions are simultaneously met:

$$I_j^{\times} \geq K_9 e_{ji}^{max} \tag{5.25}$$

$$b_i \leq 1 - K_9 \tag{5.26}$$

where K_9 is a constant set just below unity and e_{ji}^{max} is the largest $I_j^{\times}$ value seen on the inhibitory link connecting χ_j to χ_i (since the time both χ_j and χ_i have become committed). If both of these conditions are simultaneously met, it is unlikely that χ_j's pattern and χ_i's pattern overlap. This is true because, if Condition 5.25 holds, χ_j is very likely to be receiving its full input pattern. Furthermore, if Condition 5.26 holds, χ_i's pattern is probably absent.

Equations 5.23, 5.24, 5.25 and 5.26 imply that the values $I_j^{\times}$ and $I_i^{\times}$ are known locally within χ_i. Consequently, Nigrin postulates the existence of a multiplexing scheme whereby *"two independent values will be output by each cell"* [114, pg. 148]. Using such a scheme, a cell's activation and confidence values will be output by each cell within a cell assembly.

5.3.7 Learning excitatory top-down weights

The top-down (or feedback) weights, z_{ij}^{f}, are modified in an analogous fashion to the bottom-up excitatory weights. The feedback weights are initially set to zero. Thereafter, their modification is governed by the following equation:

$$\frac{\partial}{\partial t} z_{ij}^{f} = \epsilon_1 d_i(-z_{ij}^{f} + S_j d_i) \tag{5.27}$$

where S_j is the sum-normalized activity of s_j ($S_j = s_j/g_s$). Since, S_j is similar to S_{ji} (used in the excitatory bottom-up weight learning formulation), the bottom-up and top-down weights effectively become symmetric. This leads to an obvious simplification involving the use of weight sharing, which is discussed further in Appendix D.

Once a cell assembly has learned to classify a pattern, its feedback weights must be stabilized to prevent improper weight learning. Therefore, once a cell assembly has committed to a pattern and χ_i^{com} is defined as true, the feedback weights may not be modified unless χ_i responds to its full pattern. In other words, when $I_i^{\times} \geq 0.95 I_i^{max1}$. I_i^{max1} represents the largest $I_i^{\times}$ value seen on the j^{th} feedback link from χ_i. If an input cell s_j is inactive when χ_i responds to its full pattern, z_{ij}^{f} (from χ_i to s_j) is set to and frozen at zero. In this circumstance z_{ij}^{f} cannot be part of χ_i's pattern. Furthermore, let t_i denote the first time after χ_i has become committed that both the conditions $I_i^{\times} \geq 0.95 I_i^{max1}$ and $\frac{\partial}{\partial t} z_{ij}^{f} > 0$ are satisfied. Thereafter, Equation 5.27 is only followed when both of these conditions are satisfied again. This prevents the possibility of inappropriate weight decay.

The manner by which these feedback weights can be utilized to generate expectancies is discussed in Section 7.2. However, for now, the use of feedback in the chunking-out process will be discussed.

5.3.8 Chunking-out a classified pattern from $\mathcal{F}_1$

A cell assembly χ_i is considered to be *committed* if, at some time in the past, $I_i^{+} \geq 1$. In this circumstance χ_i^{com} is set to true. Later, when χ_i's pattern is present at $\mathcal{F}_1$, c_i will reach high activation levels (provided it represents the input better than any other cell assembly χ_j, and wins the competition at $\mathcal{F}_2$). If c_i remains above a certain activation level (K_{13}) for a specified time period (K_{14}), χ_i will reset its c_i cell to zero and *chunk-out* its pattern from working memory. This process is also referred to as rehearsal and works by resetting all input cells in the set T_i. At $\mathcal{F}_1$, s_j can determine if it is a member of T_i if its feedback cell's activation level, f_j, abruptly drops, which is caused by the resetting of c_i. An abrupt drop in f_j is considered to have occurred if $f_j(t - \Delta t) > 3f_j(t)$. Such a drop will enable s_j to determine its membership of T_i. Consequently, s_j will be reset to zero as part of the chunking-out process. The activation of the feedback cell is calculated as follows:

$$f_j = \sum_{i \in \mathcal{F}_2 : \chi_i^{com}} d_i z_{ij}^f \tag{5.28}$$

To allow cell assemblies coding long patterns to chunk-out, a large activation must be sustained prior to chunking-out (remember, for a time period of K_{14}). This ensures that smaller cell assemblies that represent sub-patterns of larger cell assemblies do not prematurely chunk-out their patterns. At least not until larger cell assemblies have had the chance to respond to the presence of their pattern, if indeed their pattern actually occurs. This chunking-out delay aids in solving the temporal chunking problem. Page [118, Section 7.2.3] suggests the use of variable length chunking-out delays across $\mathcal{F}_2$ which are dependent on the size of each cell assembly, while Roberts [133, Section 5.4.4] suggests a delay of a single tactus-span for processing rhythmic patterns. Nonetheless, in order to solve the temporal chunking problem and enable longer patterns to be learned and chunked-out, K_{14} must be sufficiently large.

5.4 Page's general modifications to SONNET

Over the course of Page's experimentation with SONNET (see Section 7.2), a number of modifications were introduced to generally improve the operation of the network [118, Section 6.5.3].

5.4.1 Inhibitory weight freezing conditions

Section 5.3.6 described how a pair of committed cell assemblies, χ_j and χ_i, may become non-competitive if they represent disjoint patterns. Nigrin determined that χ_j and χ_i are disjoint if χ_j responds to its complete pattern ($I_j^{\times} \geq K_9 e_{ji}^{max}$) while χ_i's pattern is absent ($b_i \leq 1 - K_9$). However, Page [118, Section 6.2.6] points out that the second condition, which tests for a pattern's absence, is unstable. Therefore, Page modified this condition to $I_i^{\times} \approx (K_1)^{|T_i|}$ in order to address this problem. The quantity $(K_1)^{|T_i|}$ is the minimum $I_i^{\times}$ value a cell assembly can reach, which is usually achieved by cell assemblies whose patterns are absent from $\mathcal{F}_1$. Consequently, a pair of committed cell assemblies, χ_j and χ_i, are now considered non-competitive if $I_j^{\times} \geq K_9 e_j^{max}$ while $I_i^{\times} \approx (K_1)^{|T_i|}$. In this situation, z_{ji}^{-} is set to and frozen at zero.

5.4.2 Using direct reset signals to chunk-out a classified pattern at $\mathcal{F}_1$

Section 5.3.8 described the chunking-out process, whereby $\mathcal{F}_1$ input assemblies that are elements of T_i are reset when a classification is performed by an $\mathcal{F}_2$ cell assembly, χ_i. An input cell, s_j, can determine whether it is a member of T_i by monitoring drops in feedback at f_j. Nigrin postulates that an abrupt drop in f_j is evidence of membership, that is, $f_j(t + \Delta t) > 3f_j(t)$.

However, Page [118, Section 6.6.2] points out that this mechanism for determining membership of T_i at s_j leads to problems when top-down priming signals are to be utilized. Instead, Page proposed that χ_i, upon performing a classification, sends direct reset signals to each input cell in its receptive field, T_i. Furthermore, this reset mechanism has advantages when event repetitions can be represented at $\mathcal{F}_1$ (see Section 6.2 for more information on handling event repetitions).

5.4.3 $\mathcal{F}_1$ dynamics

When a note onset occurs and the network is operating in event-driven mode, an attentional pulse is triggered for a brief time period (see Section 6.3). During this period, the evolution of cell activations and connection weights are enabled. However, when implementing $\mathcal{F}_1$ as an on-centre off-surround circuit (see Equation 5.1), a pattern's complete activity trace across $\mathcal{F}_1$ only emerges at the end of the attentional pulse which was triggered by the final event onset of that pattern. This situation is particularly undesirable with respect to Page's work on the generation of expectancies. Therefore, Page modified the operation of $\mathcal{F}_1$ such that changes in input cell activations are enabled for a shorter time period than cell and weight changes at $\mathcal{F}_2$ in response to the same attentional pulse. Consequently, an attentional pulse affects $\mathcal{F}_1$ and $\mathcal{F}_2$ differently. To ensure this approach operates correctly, the $\mathcal{F}_1$ parameters must be adjusted such that the ratio between input cells that respond to consecutive event onsets remain the same for an isochronously presented sequence (that is, $\omega \approx 2$).

In order to minimize computation time, Roberts simplified the $\mathcal{F}_1$ dynamics further. Briefly, at the instant an attentional pulse occurs, every $\mathcal{F}_1$ cell is scaled by the factor ω. Then, the cell that represents the event onset that just triggered the attentional pulse is set to μ. In effect, Page's modified $\mathcal{F}_1$ dynamics approach those of Roberts' simplified dynamics. Furthermore, this models the the on-centre off-surround network specified by Equation 5.1 when $g_s < K_{12}$. In this thesis we adopt Roberts approach.

5.4.4 Link confidence

Another modification that Page introduced involves the calculation of the network's link confidence, $I_{ji}^{\times}$, on committed or near committed cell assemblies. Consider the following example provided by Page [118, pg. 206–207]. Assume the sequences $\{1, 2, 3, 4\}$ and $\{1, 2, 5, 3\}$ have been learned by χ_{1234} and χ_{1253} respectively. Furthermore, assume the input sequence $\{1, 2, 3\}$ has been presented and is active at $\mathcal{F}_1$. Using the current link confidence formulation (see Equation 5.6), order information is not always adequately reflected in the overall confidence value $I_i^{\times}$. Consequently, expectancies may be incorrect. In this current example, $I_{1234}^{\times} < I_{1253}^{\times}$ when $\{1, 2, 3\}$ is presented. This is inconsistent with the fact that the start of χ_{1234}'s pattern is present while the start of $I_{1253}^{\times}$'s pattern is not. To remedy this problem, a cell assembly whose excitatory weight length

(z_i^{tot}) reaches K_Z (typically set to 0.7) will calculate its link confidence differently:

$$I_{ji}^{\times} = \begin{cases} K_1 + K_2 \min\left(1, \frac{S_{ji}}{Z_{ji}}\right) & \text{if } z_i^{tot} \; < \; K_Z \\ K_1 + K_2 \min\left(\frac{Z_{ji}}{S_{ji}}, \frac{S_{ji}}{Z_{ji}}\right) & \text{otherwise} \end{cases} \tag{5.29}$$

In the current example, $I_{1234}^{\times} > I_{1253}^{\times}$ when {1, 2, 3} is presented and Equation 5.29 is used. Therefore, event order information has a stronger influence on the assembly's bottom-up input and strengthens the quality of the pattern representations on the excitatory weights to committed or near committed cell assemblies. This leads to improved expectancy generation.

5.4.5 Omitting r_{ji} from the w_{ji}^{+} learning equation

The next modification Page introduced was to remove the learning-rate modulator, r_{ji}, from the secondary weight learning equation w_{ji}^{+} (see Equation 5.16). Page found that the inclusion of this modulator excluded partially encoded patterns from being utilized to encode similar but longer patterns once the shorter pattern had already been learned.

However, Roberts [133, Section 3.4.3] discovered that the omission of r_{ji} from Equation 5.16 led to two problems. Firstly, the omission of r_{ji} does not actually ensure (in all cases) that an uncommitted cell assembly can learn a pattern that is similar to, but longer than, its current partially encoded pattern. Furthermore, learning could terminate at these uncommitted cell assemblies. Secondly, the construction of the set T_i became unstable. Consequently, Roberts retained the learning-rate modulator in the secondary weight learning formulation and instead relied on particular choices for some parameter settings (see Section 5.5.3). In this thesis we adopt Roberts' approach.

5.4.6 Shutting-off uncommitted classification cells

Section 5.3.5 specified two conditions under which uncommitted classification cells were reset. Nigrin introduced these conditions to allow patterns that occur very infrequently, in comparison to other patterns, to be learned. This was achieved by shutting-off uncommitted cell assemblies that represent the current pattern at $\mathcal{F}_1$ relatively poorly.

However, Page discovered that it was difficult for some uncommitted classification cells, which remained active in response to the pattern at $\mathcal{F}_1$, to learn that pattern due to low M_i values. Low M_i values ultimately cause low learning-rate modulator values on the links specified by the set T_i. Consequently, Page introduced a further condition to address this problem. A cell assembly can only be reset if there exists at least one other uncommitted cell assembly, χ_l, that satisfies the following condition:

$$e_l > e_i \text{ and } M_l > K_M \tag{5.30}$$

The K_M threshold specifies how well a cell assembly must match the input at $\mathcal{F}_1$ before it is considered to represent that pattern. With this additional condition there are always cell assemblies capable of learning the input pattern at $\mathcal{F}_1$.

5.5 Roberts' general modifications to SONNET

5.5.1 Avoiding spurious termination of learning at uncommitted cell assemblies

Roberts [133, Section 3.5.1] discovered that learning at an uncommitted cell assembly (χ_i) with a good pattern representation ($z_i^{tot} > K_5 + K_6$) could terminate due to low r_{ji} values. This situation could arise if $I_i^\times$ was held at values below I_i^{max} causing M_i to be low. In turn, low M_i values result in low r_{ji} values for $j \in T_i$. Since r_{ji} is used in the calculation of I_i^{max} when $I_i^{max} \geq I_i^\times$ (see r_{0i} in Equation 5.12), I_i^{max} is prevented from decreasing. This leads to permanently low r_{ji} values for $j \in T_i$, which effectively terminates learning at χ_i.

Two circumstances lead to this situation arising. Firstly, T_i could spuriously expand when M_i was low because r_{ji} was large for $j \notin T_i$. This situation could only occur if c_i was moderately large when $I_i^\times$ was low. In this circumstance, c_i's growth should have been obstructed due to a large dilution parameter (D_i). However, because D_i could gradually decrease towards $I_i^\times$, c_i was able to increase further, thus causing a spurious expansion of T_i. To hinder D_i from decreasing to $I_i^\times$ in this situation, the term r_{0i} was introduced into the D_i formulation:

$$\frac{\partial}{\partial t} D_i = \epsilon_1 \epsilon_2 r_{0i} c_i \left[-D_i + I_i^\times \right] \tag{5.31}$$

Secondly, Roberts discovered that Page's $I_{ji}^\times$ modification (presented in Section 5.4.4) introduced a discontinuity into the $I_i^\times$ formulation at the point when z_i^{tot} reaches K_Z. This discontinuity could cause $I_i^\times$ to suddenly drop as z_i^{tot} increased. This resulted in a low M_i value and consequently, low r_{ji} values for $j \in T_i$. Thus learning at χ_i would terminate. To remedy this problem, Roberts introduced the term f_i into the $I_{ji}^\times$ formulation. This permitted $I_i^\times$ to decrease only gradually in this situation, thus removing the possibility of spurious sudden drops in $I_i^\times$:

$$I_{ji}^\times = K_1 + K_2 \min \left[\left(\frac{Z_{ji}}{S_{ji}} \right)^{f_i}, \frac{S_{ji}}{Z_{ji}} \right] \tag{5.32}$$

$$f_i = \frac{1}{1 + e^{100(K_Z - z_i^{tot})}} \tag{5.33}$$

5.5.2 Limiting inhibition at uncommitted cell assemblies

Section 5.3.5 described how a ceiling of ($K_7 e_i^{big}$) was placed on uncommitted cell assemblies to avoid a dramatic slow-down in learning times. However, Roberts discovered that e_i^{big} could be derived from a cell assembly (χ_j) with large excitatory weights ($z_j^{tot} > K_5 + K_6$) but with a low M_j value. Furthermore, χ_j could have a large e_j activation if the start of its pattern is present at $\mathcal{F}_1$. In this situation, since M_j is low, χ_j would be unable to learn from the current input pattern at $\mathcal{F}_1$. Consequently, the inhibition from e_j to other cells assemblies only serves to obstruct learning at those cell assemblies.

To address this problem, Roberts introduced a modification whereby e_i^{big} may only be derived from cell assemblies that are capable of learning patterns that are present at $\mathcal{F}_1$. The set Q was introduced, which consists only of cell assemblies (χ_j) that satisfy the following criteria:

1. χ_j must be uncommitted.

2. $z_j^{tot} \leq K_5 + K_6$ or $M_j > K_M$

e_i^{big} was then computed as follows:

$$e_i^{big} = \max_{j \in Q}(e_j) \tag{5.34}$$

Furthermore, only cell assemblies in the set Q could have their maximum inhibitory input limited to $(K_7 e_i^{big})$. This facilitated the learning of patterns at a reasonable speed, even if the input collection consisted of sequences with many similar patterns.

5.5.3 Exploiting cell assemblies with partial pattern representations

Roberts [133, Section 3.5.3] showed that uncommitted cell assemblies, which have formed pattern representations that are no longer useful, could be exploited to learn different, but similar patterns. This could be achieved by setting certain parameters in a way that enables z_i^{tot} to decrease below $K_5 + K_6$ (the point at which the learning rate modulator comes into effect) as rapidly as possible. This could be achieved by setting $A = 1$, $B = 2$, $v_1 = 3$, $v_2 = 30$ and $K_7 = 50$.

5.5.4 Modifying D_i

Section 5.5.1 described how Roberts introduced the learning-rate modulator into the D_i equation. This change made it more difficult for D_i to decrease when $I_i^{\times}$ was low, thus removing the problem of spurious T_i expansion. However, during exploratory time-driven simulations, Roberts discovered that D_i could undesirably decrease when $z_i^{tot} \leq K_5 + K_6$, which was *before* the learning-rate modulator could influence processing. Consequently, the excitatory weights could modulate between different patterns that were presented at $\mathcal{F}_1$. Thus, a cell assembly's pattern representation could be unstable at this early stage of learning. To overcome this problem, D_i was modified such that it could increase more easily than it could decrease. This was achieved by strengthening the influence that c_i has on the evolution of D_i. That is, the term c_i^2, rather than c_i, controlled D_i's rate of change:

$$\frac{\partial}{\partial t} D_i = \epsilon_1 \epsilon_2 r_{0i} \max(c_i^2, K_D) \left[-D_i + I_i^{\times}\right] \tag{5.35}$$

where K_D is a small constant which specifies a floor value, allowing D_i to decrease gradually when c_i is very low.

5.5.5 A new learning-rate modulator formulation

The formulation presented below is a modified version [133, Section 6.7] of Nigrin's original specification [114, Section 3.14.4]. This version removes discontinuities from the original formulation which caused problems with learning after a cell assembly has already achieved a fair representation of an input pattern.

$$r_{ji} = \left[(T_{ji})^{(1-T_{ji})M_i}(1-T_{ji})^{T_{ji}(1-M_i)}\right]^{f_i} \tag{5.36}$$

$$T_{ji} = 0.1 + \frac{0.8}{1 + exp\left[-4f_i\left(\frac{w_{ji}}{w_{0i}} - \frac{1}{K_3}\right)\right]} \tag{5.37}$$

$$f_i = \begin{cases} 20\sqrt{z_i^{tot} - K_5} & \text{if } z_i^{tot} > K_5 \\ 0 & \text{otherwise} \end{cases} \tag{5.38}$$

where T_{ji} quantifies whether or not a bottom up link is a member of the set T_i and f_i enables the formulation to model the behaviour of Nigrin's original specification.

5.6 Summary

This chapter explained the fundamental concepts behind SONNET and presented Nigrin's original SONNET specification (see Nigrin [114, Appendix C] for a condensed version of this specification). Further general modifications made by Page and Roberts were also discussed. The next chapter will discuss more advanced SONNET topics, including: how event repetitions at $\mathcal{F}_1$ can be handled, attentional signal control and the manner by which SONNET modules can be cascaded to form a hierarchy.

Chapter 6

Advanced SONNET Topics

6.1 Introduction

In Chapter 5 the fundamentals of SONNET's theory and its formal specification were presented. In this chapter, the following advanced topics will be discussed:

1. The means by which SONNET can process input sequences containing repeated event types using multiple interacting presynaptic links.

2. How a non-specific attentional control signal can be utilized to enable SONNET to become either event-driven or time-driven.

3. How SONNET modules can be cascaded to form a single-dimensional network hierarchy.

The topics covered in this chapter will prove particularly salient when considering the new work presented in Part III of this thesis.

6.2 Handling event repetitions

Since temporal sequences may contain multiple occurrences of the same input event (this is particularly true of musical sequences), the ability to represent, learn and recognize patterns containing such event repetitions is essential. Moreover, this facility is a pre-requisite for new work on *phrase repetitions* introduced in Chapter 8. In the meantime, however, this section describes work by Nigrin [114], Page [118] and Roberts [133] that shows how event repetitions can be processed.

6.2.1 Storing event repetitions at $\mathcal{F}_1$

Employing a single input cell and using the magnitude of its activation level to represent multiple occurrences of the same event type is clearly unfeasible. The use of a solitary activation level leads to difficulties in determining the number of event repetitions that have occurred in addition to retaining order information. Therefore, the use of multiple storage locations must be used to unambiguously represent each repetition of the same event type.

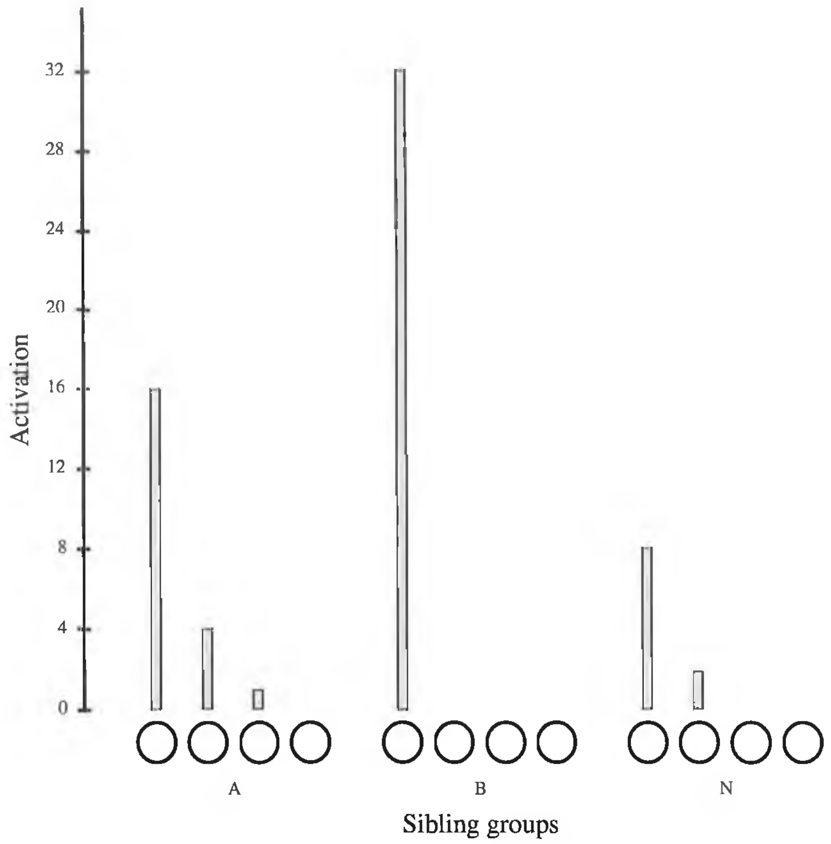


Figure 6.1: The activity trace of $\mathcal{F}_1$ in response to the sequence of letters {B, A, N, A, N, A}. In this example, $\mu = 1$, $\omega = 2$ and $N_s = 4$. Page [118, Section 4.10.2] refers to the problem of representing repeated items as the *"banana problem"* due to the difficulties in processing repetitions of the letters A and N.

Each repetition of an event could be funnelled into one of a set of cells that represent the event type in question, with each storage location tagged with a number indicating the occurrence order. For example, cells representing the item A could be indexed A_1, A_2, A_3 and A_4, where the first occurrence of A is funnelled to A_1, the second occurrence of A is funnelled to A_2 and so forth. However, using this approach leads to practical difficulties. For example, suppose the patterns {A, B} and {B, A} have been learned by χ_{AB} and χ_{BA} respectively. Now, when the sequence {A, B, B, A} is presented, the cells A_1, B_1, B_2 and A_2 should activate. Although χ_{AB} will recognize its pattern at $\mathcal{F}_1$, χ_{BA} will not. This is because $\mathcal{F}_1$ represents a list in which the occurrence of the pattern {B, A} is represented by input cells that indicate they are the second occurrence of these items in a list. In other words, since each storage location is tagged with an index which represents the occurrence number, cells that represent the occurrence of items A or B are not completely synonymous and will not be treated as such by $\mathcal{F}_2$ cell assemblies. In effect, χ_{AB} has actually learned the sequence that activates the cells A_1 and B_1, whereas χ_{BA} has actually learned the sequence that activates the cells B_1 and A_1.

Other STM mechanisms have been developed that use multiple storage locations to represent event repetitions. These include tapped-delay lines [66, Section 13.2], gamma memory [165] and shift registers [168, 169]. However, these methods tag each storage location with either an onset time or an index indicating occurrence number which, as we have just seen, is not practical. Nigrin [114, Section 6.1] and Page [118, Section 7.2.1] also discuss the disadvantages of these approaches and conclude that different, structurally-distinct storage locations must represent each occurrence of the same type of event without regard to other occurrences of that event.

When distinct storage locations are used within the SONNET framework, $\mathcal{F}_1$ is divided into N_g sibling groups — one for each event type. Furthermore, each sibling group contains N_s siblings (input assemblies), which determine the maximum number of event repetitions that each sibling group may concurrently store. Figure 6.1 shows how the sequence {B, A, N, A, N, A} can be represented at $\mathcal{F}_1$. In particular, three occurrences of the item A and two occurrences of the item N are stored. Any of the siblings representing the items A or N may have activated in response to their occurrence. In other words, siblings within the same sibling group are completely synonymous with one another.

The choice of storage location to be used in each sibling group may employ the use of a repeat selector such as those described by Nigrin [112] and Bradski et al. [11], whereby a combination of randomly varying connection strengths and a *winner-take-all* preprocessor is used to choose an input assembly to activate in response to an event occurrence. However, the behaviour of these circuits can be modelled by simply selecting the next available inactive input assembly. In any case, this scheme allows the unambiguous representation of multiple occurrences of the same input event at $\mathcal{F}_1$.

6.2.2 Using multiple links to connect each $\gamma - \chi$ pair

Event repetitions are represented at $\mathcal{F}_1$ by structurally distinct input assemblies within the event type's sibling group. Since any storage location within a sibling group may activate in response to each repetition, the same input sequence can be represented by numerous activity traces across $\mathcal{F}_1$. Therefore cell assemblies are required to recognize patterns that contain event repetitions regardless of which siblings within a particular sibling group activate. For example, Figure 6.2 shows two possible activity traces for the sequence of notes {A, B, B, A}. Consequently, a cell assembly that has learnt this pattern, χ_{ABBA}, must be capable to recognizing when its pattern occurs across $\mathcal{F}_1$, regardless of which input assemblies happen to activate in response to its input sequence. Nigrin [114, pg. 242] refers to this issue as one aspect of the *synonym problem* where input assemblies within the same sibling group are considered *exact* synonyms for one another.

A separate cell assembly at $\mathcal{F}_2$ could be used to encode each possible activity trace that may occur across $\mathcal{F}_1$ in response to a pattern. However, the combinatorial explosion that would result from learning all of these permutations would lead to inefficient, bloated networks. For example, an input field containing a single sibling group of twenty siblings could represent a pattern of five onsets in $1,860,480$ (${}^{20}P_5$) different ways. In fact, this is the configuration used for the CBR experiments described in Chapter 13. It is clearly impractical for this number of cell assemblies to learn a single pattern consisting of five onsets. Instead, Nigrin [112] introduced a solution to

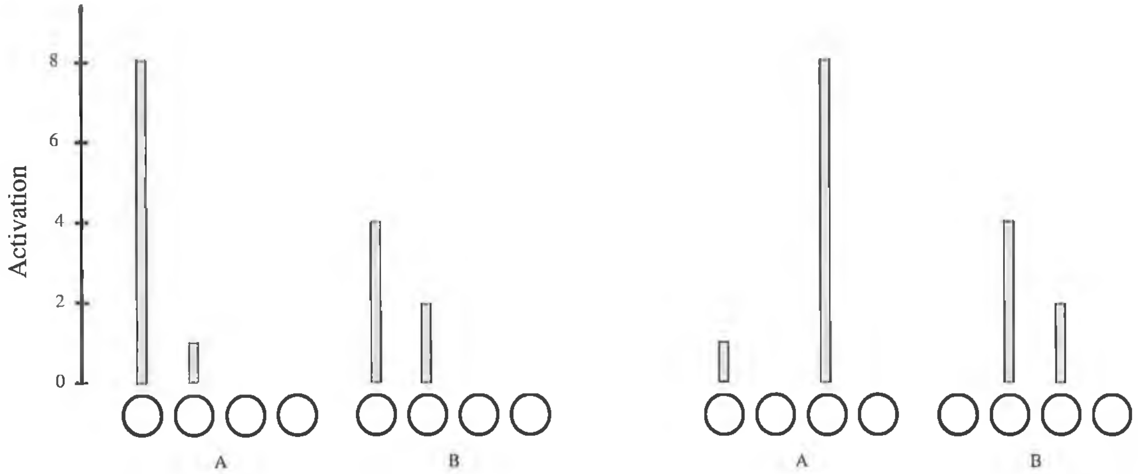


Figure 6.2: Two examples of activity traces that may occur across $\mathcal{F}_1$ in response to the sequence {A, B, B, A}. In this example, $\mu = 1$, $\omega = 2$ and $N_s = 4$. $\mathcal{F}_2$ must be capable of recognizing both traces as the same pattern.

this problem whereby a pattern can be recognized by a single cell assembly regardless of which particular input assemblies may represent the pattern at any particular point in time. Rather than using multiple cell assemblies to learn all possible permutations, each γ-χ pair is connected by *multiple bottom-up links*[1]. Although other researchers have studied the use of multiple links, Nigrin's mechanism is more robust because set storage locations are not required.

6.2.3 Presynaptic interactions between links

Nigrin [112] first described the manner by which multiple links could interact to allow patterns containing event repetitions to be recognized irrespective of which storage locations, within a sibling group, are used. The structure of these links and the nature of their interactions is described next.

Link structure

Figure 6.3 shows the link architecture between input assemblies at $\mathcal{F}_1$ (that are within the same sibling group) and a single cell assembly at $\mathcal{F}_2$. The number of links between each $\gamma - \chi$ pair is specified by N_l. The term *link cluster* is introduced here to describe the set of N_l links between each $\gamma - \chi$ pair. Each individual link is associated with two areas of locality known as *relay cells*; a lower relay cell (LRC) and an upper relay cell (URC). LRCs are labelled as l^k_{mj} and URCs are labelled u^k_{mj}, where j denotes the input assembly's group, m denotes the index of the input assembly within the group and k denotes the link's strength. In Figure 6.3, only a single sibling group is used, therefore the index j is dropped. The strength of each link is defined by the extent of

[1] Remember, in Chapter 5, the term γ_j was introduced to collectively refer to each set of cells that process a particular event occurrence.

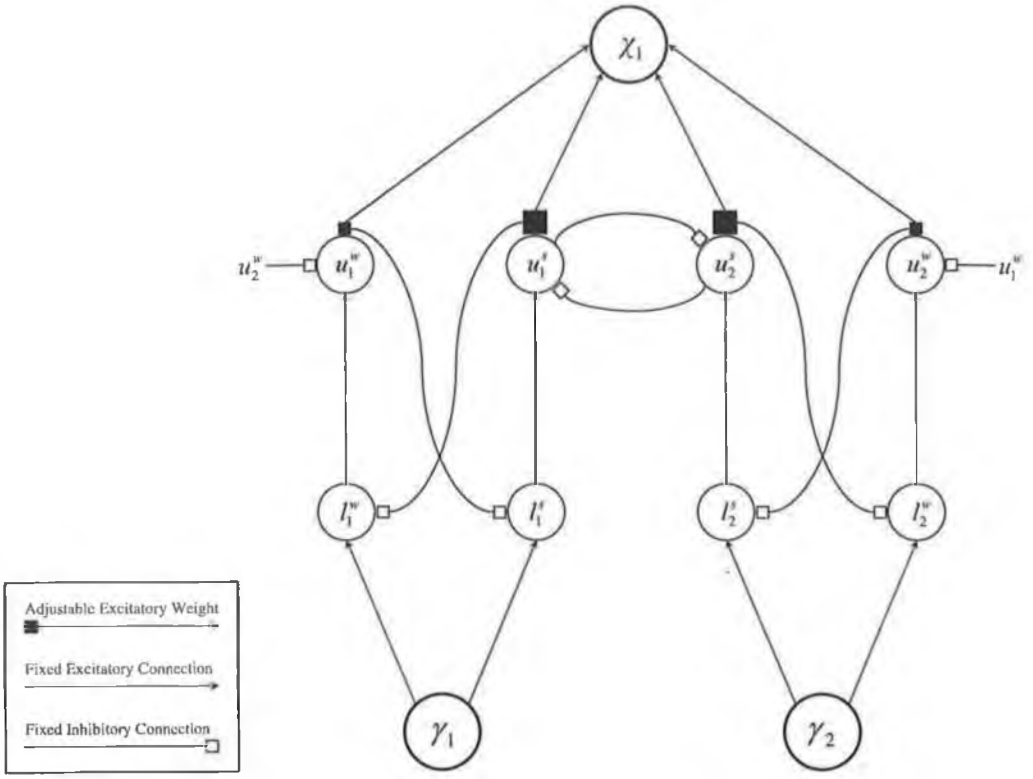


Figure 6.3: Multiple links connecting each γ-χ pair, where γ_1 and γ_2 are from the same sibling group. Note the relay cells and in particular the inhibitory connections between them. The manner by which these inhibitory links interact enables event repetitions to be handled properly. In this example, $N_s = 2$ and $N_l = 2$.

the excitatory weight associated with that link, with $k = 1$ indicating the strongest link and $k = N_l$ indicating the weakest link. For simplicity, in all the examples and diagrams in this section, the strongest link is labelled with s and the weakest link with w. Furthermore, note that the number of links (N_l) may be less than the number of siblings in each group (N_s). In this circumstance it is possible that some signals may not be transmitted to $\mathcal{F}_2$ at all.

When γ_m activates, its signal is sent to all the LRCs within each link cluster projecting to $\mathcal{F}_2$. Each LRC will then attempt to forward this signal to its associated URC. An URCs output is gated by the link's excitatory weight before reception at χ_i and every LRC within its link cluster (apart from the LRC on its own link). Each learned weight represents a particular occurrence of an item within the cell assembly's pattern. The weight on the strongest link represents the first item in the pattern, while the weakest link represents the last item in the pattern. It is important to note that, all link clusters emanating from a particular sibling group, that share the same terminating cell assembly have an identical set of excitatory weights. It is this redundancy that enables the link architecture to operate properly. Furthermore, this redundancy can be exploited to model this neural circuit in an efficient manner (see Section 6.2.5). Additionally, all links which originate from the same sibling group, terminate at the same cell assembly and have identical excitatory weight values will also compete. The manner by which competition operates within link clusters and between link clusters will be discussed next.

Link competition

When an input assembly activates (for example, γ_1 in Figure 6.3), links within the cluster $\gamma_1 - \chi_1$ will compete to transmit this signal to χ_1. Subsequently, when γ_2 activates, links within the link cluster $\gamma_2 - \chi_1$ will also compete to transmit the signal from γ_2 to χ_1. Additionally, the link cluster $\gamma_1 - \chi_1$ will compete with the link cluster $\gamma_2 - \chi_1$. However, this competition between different link clusters only takes place between links that code for the same list position in χ_1's pattern. That is, links which have identical excitatory weight values.

This competition is facilitated by *presynaptic inhibitory connections* that are arranged as shown in Figure 6.3. Two conditions specify the manner by which presynaptic inhibitory connections may compete:

1. At most, one link from each link cluster may be active at any given time. That is, links within a cluster are mutually inhibitory. Therefore, each link cluster will represent a single occurrence of an item in a cell assembly's pattern. To achieve this condition, each URC sends an inhibitory signal (which is gated by its excitatory weight) to every LRC within its link cluster (apart from the LRC on its own link). Consequently, links with larger excitatory weights will initially have a competitive advantage over links with smaller excitatory weights. For example, the strongest link within $\gamma_1 - \chi_1$ will always overpower the weaker link, provided no other input assemblies within γ_1's sibling group are active.

2. Links that (1) originate from different siblings within the same sibling group, (2) terminate at the same cell assembly and (3) code for the same list position in a cell assembly's pattern (that is, have identical excitatory weights) should not activate simultaneously. That is, these links are also mutually inhibitory. For example, the strongest links within $\gamma_1 - \chi_1$ and $\gamma_2 - \chi_1$ should not be simultaneously active. To achieve this condition, such links compete via their URCs by sending inhibitory signals to each other. Consequently, links that respond to older events will have a greater competitive advantage over links that respond to more recent events. This is due to the short-term storage of order information using a primacy gradient (see Section 5.3.1). For example, if γ_1 activates before γ_2, the strongest link within $\gamma_1 - \chi_1$ will overpower the strongest link within $\gamma_2 - \chi_1$.

Since it is the links that terminate on the same cell assembly that compete, this form of competition is called *intra-cell competition*.

A highly contrast-enhancing type of competition is assumed to take place between competing links. This avoids excessive run times and enables a winning link to completely shut-off its competitors. Additionally, identical learning must take place between all link clusters that share the same sibling group and terminate at the same cell assembly. Section 6.2.5 will show how this behaviour can be modelled.

Example

The following example, with reference to Figures 6.4 and 6.5, illustrates the behaviour of these link interactions in achieving the recognition of phrases containing event repetitions. Suppose the pattern {A, A} has been learned by χ_1 and that γ_1 and γ_2 each respond to occurrences of the item A. χ_1 must be capable of recognizing its pattern regardless of the order in which γ_1 and γ_2 activate in response to {A, A}. Suppose, the item A occurs, and activates γ_1. This signal is propagated to the LRCs (l_1^s and l_1^w) and relayed to the URCs (u_1^s and u_1^w). At this point, the signals are gated by the excitatory weights on each link and relayed back to the LRCs via the presynaptic inhibitory connections. In this circumstance, despite receiving the same input signal, the strongest link will overwhelm the weaker link due to its larger excitatory weight. This results in the weaker link being suppressed. Consequently, the strongest link will transmit the signal at γ_1 to χ_1 (see Figure 6.4).

Now, when γ_2 fires in response to a second occurrence of A, the signal from γ_2 is propagated to the LRCs (l_2^s and l_2^w) and relayed to the URCs (u_2^s and u_2^w). In this circumstance, u_1^s will suppress u_2^s because both links code for the same position in χ_1's pattern and γ_1 is transmitting a stronger signal than γ_2 (due to the decreasing activity pattern at $\mathcal{F}_1$). Consequently, the weakest link in the link cluster from γ_2 to χ_1 will transmit its signal because it receives no inhibition from the stronger link, which has been suppressed. In this way, χ_1 recognizes its pattern across $\mathcal{F}_1$ (see Figure 6.5).

Conversely, if γ_2 had activated first, followed by γ_1, similar behaviour would enable χ_1 to recognize its pattern. Therefore, regardless of the permutations in which the input assemblies activate, χ_1 will recognize if its pattern is active at $\mathcal{F}_1$.

6.2.4 Recognizing embedded patterns

Roberts [133] discovered that *embedded* patterns could not be reliably recognized using the current rules governing presynaptic inhibitory interactions. Suppose the pattern {A, A, B} has been learned by χ_1 in Figure 6.6. Furthermore, assume γ_1, γ_2 and γ_3 respond to occurrences of the item A and that γ_4 (not shown in Figure 6.6) responds to occurrences of the item B. In this example, two links connect each $\gamma - \chi$ pair ($N_l = 2$). With this setup, χ_1 will be unable to recognize its pattern if the input sequence {A, A, A, B} is presented at $\mathcal{F}_1$. In other words, when {A, A, B} is embedded within {A, A, A, B}.

Just after {A, A} has been presented the behaviour of the circuit is identical to that described in Section 6.2.3. However, when the third occurrence of A is presented, activating γ_3, no signal from γ_3 will be propagated to χ_1 because u_3^s and u_3^w will be inhibited by u_1^s and u_2^w respectively. Consequently, χ_1 is unable to respond to the third occurrence of the event A. Now, when γ_4 activates in response to the occurrence of item B, γ_4 will propagate its signal to χ_1 via its strongest link in the usual way. Since γ_4 is in a separate sibling group, it is not involved in the competition occurring between the links from γ_1, γ_2 and γ_3 to χ_1. At this point in time χ_1 will be responding to the pattern specified by γ_1, γ_2 and γ_4 — {A, A, –, B}, rather than the pattern specified by γ_2, γ_3 and γ_4 — {A, A, B}. Consequently, the match between χ_1's excitatory weights and normalized

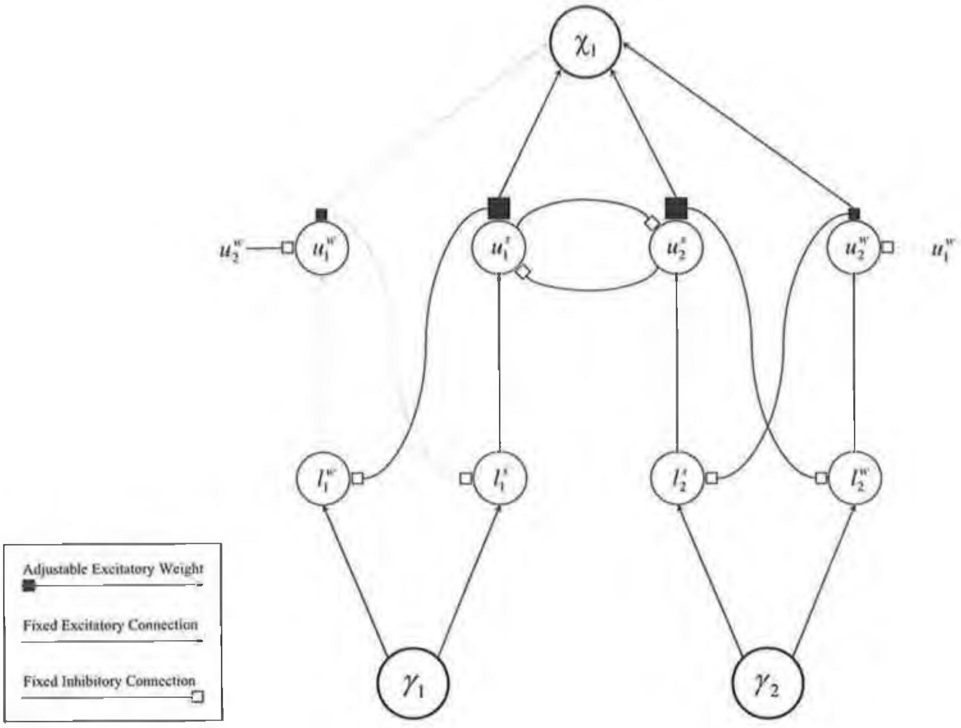


Figure 6.4: In response the first occurrence of the item A, the strongest link from γ_1 to χ_1 will transmit the input signal. This is due to its larger excitatory weight. Note, the grey links indicate that no signal is currently being carried on that link, which is due to its suppression.

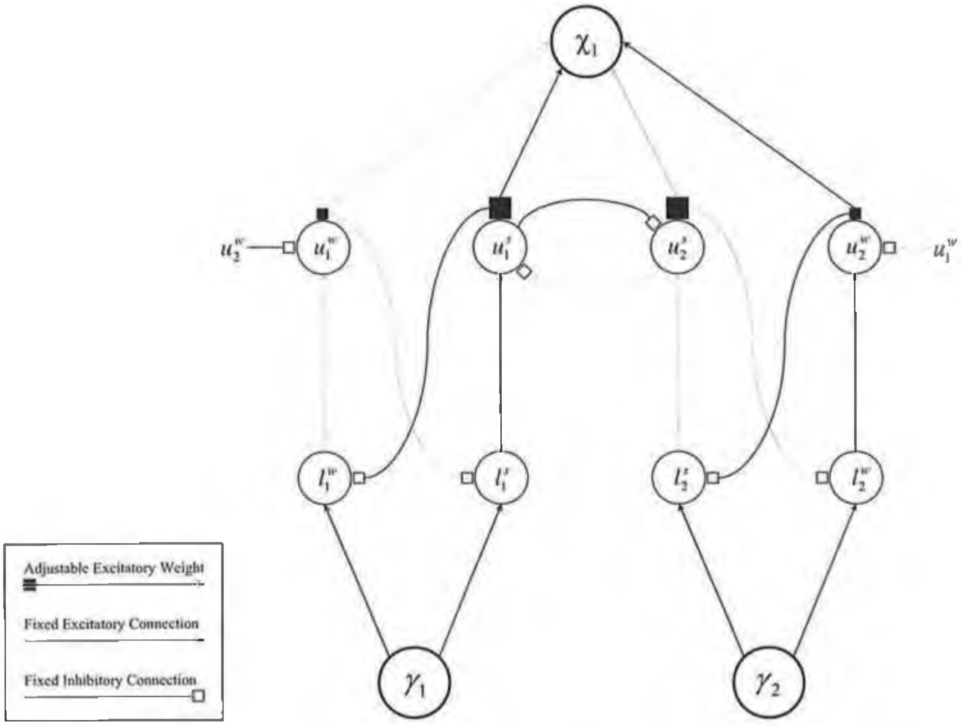


Figure 6.5: In response the second occurrence of the item A, the weakest link from γ_2 to χ_1 will transmit the input signal. This is due to the suppression of the strongest link by u_1^s.

inputs is low, thus χ_1 fails to recognize the presence of its pattern at $\mathcal{F}_1$. A modification is required to enable u_2^s and u_3^w to respond to γ_2 and γ_3 respectively, thus enabling χ_1 to recognize its pattern.

Crucial to solving this problem is an observation made by Roberts regarding the onset of the third event A. When γ_3 is activated, its excitatory signal is propagated to u_3^s and u_3^w. Both u_3^s and u_3^w also receive inhibitory signals from u_1^s and u_2^w respectively. Roberts observed that all URCs from the link cluster $\gamma_3 - \chi_1$ (u_3^s and u_3^w) receive *simultaneous sustained excitatory signals from* γ_3 *and sustained inhibitory signals from* u_1^s *and* u_2^w. The use of the word *sustained* is important because it is possible for a relay cell to temporarily receive inhibition and excitation at the same time, which would not precipitate a reset. The proposed mechanism involves the use of a *relay-cell reset* mechanism whereby URCs that receive sustained excitatory and inhibitory signals cause the URCs that provided the inhibitory signals to be reset to zero and remain dormant for as long as their corresponding input assemblies at $\mathcal{F}_1$ remain active. The inhibitory links from u_3^s to u_1^s and from u_3^w to u_2^w can be utilized to send this reset signal. This is similar to the manner by which feedback links are used to send direct reset signals to $\mathcal{F}_1$ input assemblies in response to a chunk-out.

In Figure 6.7 this mechanism would cause the reset of u_1^s and u_2^w. In the case of γ_1, all bottom-up links to χ_{AAB} must be shut-off. Otherwise, the signal from u_1^w would suppress u_3^w and the desired behaviour would not be obtained. Therefore, in this circumstance, the relay-cell reset mechanism is extended so that the URC associated with the the strongest link can additionally propagate the reset signal to each of the LRCs within its link cluster. In effect, the entire link cluster is reset for as long as γ_1 remains active. Therefore, in this example l_1^w will be shut-off and remain inactive for as long as it receives input from γ_1. Consequently, no inhibitory signals will be sent from the link cluster connecting γ_1 to χ_1. Furthermore, since u_2^w will be reset by u_3^w, u_2^s will no longer be inhibited by either u_2^w nor u_1^s. Therefore, u_2^s will propagate its signal to χ_1. Since the URCs in the link cluster from γ_3 are no longer receiving sustained inhibitory signals, they can compete in the usual way. Since u_2^s will inhibit u_3^s, u_3^w will win the competition to send its signal to χ_1. In this way the pattern {A, A, B} will be recognized by χ_1 even though it is embedded within the pattern {A, A, A, B}.

The relay-cell reset mechanism may only be invoked in accordance with the following rules:

1. The mechanism should only be invoked by a cell assembly, χ_i, if it does not match the current input at $\mathcal{F}_1$ well enough. That is, if $M_i < K_M$. For example, the same situation can arise as shown in Figure 6.6 in response to the input pattern {A, A, B, A}. However, in this circumstance, χ_1 would have already recognized its pattern at $\mathcal{F}_1$ at the instant the item B occurred. Consequently, in this situation the relay-cell reset mechanism should not be invoked.

2. Only cell assemblies that have a good pattern representation should be allowed to invoke this mechanism. Cell assemblies are considered to have a good pattern representation if they have relatively large excitatory weights. That is, if $z_i^{tot} > K_{embed}$. Typically, K_{embed} is set between 0.3 and 0.5.

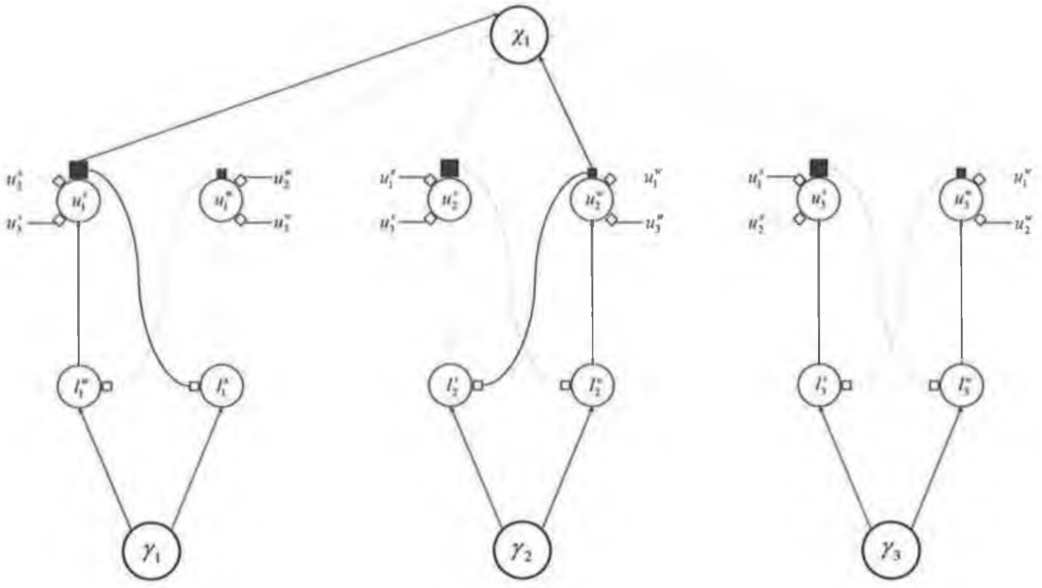


Figure 6.6: The state of the inhibitory links in response to the sequence {A, A, A} before the relay-cell reset mechanism has been invoked.

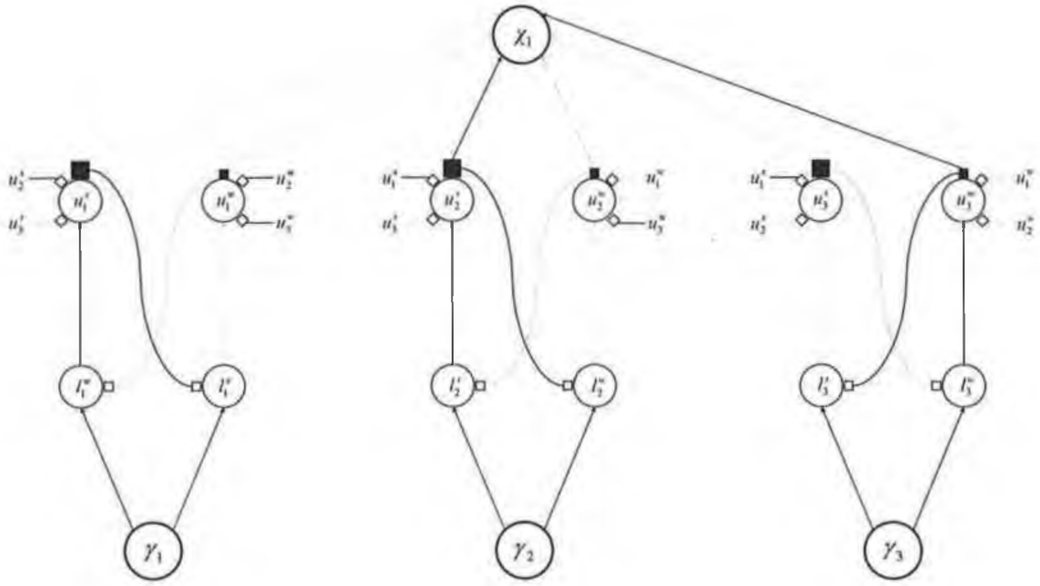


Figure 6.7: The state of the inhibitory links in response to the sequence {A, A, A} after the relay-cell reset mechanism has been invoked.

3. Links that are excluded from T_i may not inhibit links that are included in T_i. This enables cell assemblies that contain fewer significant links (from a particular input assembly) than there are actual links (specified by N_l) to recognize their pattern when they are embedded within larger patterns. Roberts [133, Section 4.3.4] referred to these types of patterns as *short embedded patterns*. Furthermore, this condition emphasizes the shift in definition of the set T_i. Instead of T_i specifying the set of *input assemblies* that form χ_i's pattern, T_i now specifies the set of *links* that form χ_i's pattern. Nonetheless, T_i is calculated in exactly the same way as before. With this condition, χ_{AB} will be able to recognize its pattern even if it is embedded within the pattern $\{$A, A, B$\}$, when $N_l \geq 2$.

When a link is reset using the relay-cell reset mechanism, the signal from this link, which is in the set T_i, will cause an abrupt drop in b_i. Consequently, c_i will be reset. This prevents c_i from continually increasing when χ_i is responding to different embedded patterns.

6.2.5 Modelling the multiple interacting links

As Section 6.2.3 has already pointed out, each link cluster from the j^{th} sibling group to the i^{th} cell assembly contains an identical set of excitatory weights. For example, the strongest link in each of these link clusters (that is, $k = 1$) will have identical excitatory weight values. Although Nigrin [114, Section 6.9] explains how identical learning can occur at each of these clusters, Page [118, Section 7.2.2] takes advantage of this redundancy and introduces an optimization that can be utilized when implementing the link structure and the corresponding inhibitory interactions.

Indexing scheme

Each $\gamma - \chi$ pair is connected via N_l bottom-up links. Therefore, there are $N_l N_s$ bottom-up links connecting each sibling group to a single cell assembly. However, there are only a maximum of N_l different link strengths. To take advantage of this redundancy, the following indexing scheme is employed:

- $\mathcal{F}_1$ is divided into N_g sibling groups. Each group is indexed by j within the range $[1 \leq j \leq N_g]$.

- Each sibling group contains N_s input assemblies. Each sibling is indexed by m within the range $[1 \leq m \leq N_s]$.

- All link clusters that originate from the j^{th} sibling group at $\mathcal{F}_1$ and terminate at the same cell assembly at $\mathcal{F}_2$, share a set of N_l excitatory weights. Each of these excitatory weights is indexed by k, which is within the range $[1 \leq k \leq N_l]$.

- $\mathcal{F}_2$ contains N_2 cell assemblies. Each cell assembly is indexed by i within the range $[1 \leq i \leq N_2]$.

Using this indexing scheme, the excitatory weight from the j^{th} sibling group to the i^{th} cell assembly on the k^{th} strongest link is referenced by z^{+}_{kji}. Other values are referenced similarly: w^{+}_{kji}, r_{kji}, Z_{kji}, S_{kji}, $I^{\times}_{kji}$ and z^{J}_{kij}.

Rank

Each input assembly has an associated value called a *rank* (introduced by by Page [118, Section 7.2.2]), labelled R_{mj} in the range $[0 \leq R_{mj} \leq N_s - 1]$. A cell assembly's rank equals the number of other input assemblies within its sibling group with larger activation levels. In the case of input assemblies that have the same activation levels, which occurs for inactive input assemblies, the tie is broken arbitrarily. The cell with the largest activation will have $R_{mj} = 0$ and will have its activation gated by the excitatory weight on the strongest link (z^{+}_{1ji}). The cell with the smallest activation will have $R_{mj} = N_s - 1$ and will have its activation gated by the excitatory weight on the weakest link ($z^{+}_{N_l ji}$). This scheme minimizes computation time because the link interactions do not have to be explicitly simulated. Furthermore, the computation of the rank values are the only extra calculation required (apart from working out the relay-cell resets).

Relay-cell resets

When the relay-cell reset mechanism is invoked by χ_i and shuts-off the link cluster associated with the strongest excitatory weight, the signals from the other siblings within the group will shift to next strongest excitatory weight associated with their link cluster. For example, the sibling with $R_{mj} = 1$ will now gate the strongest link in its cluster and the sibling with $R_{mj} = N_s$ will now gate the weakest link in its cluster. If another relay cell reset occurs, the sibling with $R_{mj} = 2$ will now gate the strongest link in its cluster and so forth. Generally, the strongest link will be gated by the sibling whose R_{mj} value equals the number of shut-off links to χ_i from the j^{th} sibling group. This models the links' interactions described in this section whilst at the same time minimizing computation time.

6.3 Attentional subsystem and timing considerations

A *non-specific attentional control signal* can be used to restrict the evolution of cells and weights within a SONNET module. The signal is considered as non-specific because it affects every cell and weight within the network identically, without targeting any specific network component. Network behaviour can be altered dramatically depending on how the attentional signal is configured. This section considers two possibilities that are pertinent to the simulations described in later chapters.

6.3.1 Time-driven mode

The SONNET network described so far is considered to be in *time-driven* mode because its cells and weights are continually changing, irrespective of whether events occur or not. That is, the attentional signal is always on. Networks that operate in this way are sensitive to the time interval between consecutive event onsets. The primary consequences of this are:

- Not only is the order in which the events of a sequence occur stored, but the rhythm of a sequence (in terms of IOIs) is also stored. Remember, once activated, input cell activations continually increase at a constant rate ω until they are reset. Consequently, the ratio between input cell activations that respond to consecutive event onsets will represent a sequence's rhythm. Furthermore, since a cell assembly's excitatory weights converge to $\mathcal{F}_1$'s activation pattern, relative weight strength will store these rhythmic patterns in LTM. In Section 7.3, work by Roberts will show how IOIs can be represented at $\mathcal{F}_1$ by relative cell activations, and subsequently learned by $\mathcal{F}_2$ cell assemblies.

- Both sequential regularities *and* rhythmic regularities will affect the patterns and segmentations formed. For example, relatively long IOIs allow SONNET a greater length of time to respond to its current input pattern at $\mathcal{F}_1$. As a consequence, many of the segments formed will end in relatively long IOIs. This models the *gap* principle of rhythm perception (see Chapter 2).

- A pattern that has been learned using a particular presentation rate cannot be recognized in a robust manner when the same pattern is presented using a different presentation rate. This is because particular parameter choices for Equation 5.1 lead to a particular value for ω. Therefore, if the presentation rate changes, ω changes too. Moreover, the activation ratios between input cells that respond to consecutive onsets will differ at various presentation rates. In other words, the rhythmic representations at $\mathcal{F}_1$ are highly dependent on absolute time intervals. In musical terms, this problem is referred to as tempo invariance. Nonetheless, Roberts [133, Chapter 8] shows how a beat-tracking signal can be used to modulate the $\mathcal{F}_1$ parameters (see Section 5.3.1) to allow tempo invariant pattern recognition. This enables identical rhythms, which are presented at different speeds, to be recognized as the same rhythm.

Although time-driven networks have proven useful for modelling musical rhythms, they are not suitable for use at more abstract layers, when SONNET modules are cascaded to form a hierarchy. Therefore, these modules should operate in event-driven mode.

6.3.2 Event-driven mode

Sometimes, it is useful to enable SONNET to operate in *event-driven* mode. In this mode, the network pays attention and responds directly to event occurrences by firing the non-specific attentional control signal. Events are considered to have occurred when an input assembly has been activated, whether this is directly by an input item or by a pattern classification (in the case of a hierarchy). When an event occurs at $\mathcal{F}_1$, the attentional signal activates for a brief time period (T_{attn}, which is typically set to between 0.15 seconds and 0.2 seconds), during which cells and weights are allowed to evolve at $\mathcal{F}_1$ and $\mathcal{F}_2$. After this time period has elapsed, the attentional signal becomes inactive again, causing network evolution to pause until the next event occurs. In effect, event-driven SONNET modules consider input sequences as isochronous streams of input events, regardless of their rhythms or presentation rates. Consequently, event-driven networks are tempo invariant.

These properties afford event-driven networks with a great deal of flexibility and allow them to achieve the following:

- Event-driven networks are well suited to discovering sequential regularities. For example, Roberts [133, Chapter 4] shows how regularities in sequences of quantized IOI's can be discovered using such a configuration.

- It is also possible to enable the attentional pulse to fire when an event is expected to occur, whether or not one actually occurs. Page [118, Section 6.3] describes how a *neural metronome* can be used to enable the rhythm of a melody to be represented (input cells are gated by the neural metronome), in addition to allowing enhanced learning for notes with relatively long durations (weights are also gated by the neural metronome). However, unless the metronome tracks beats at a sufficiently low metrical level (which is difficult [133, pg. 122]), it is possible that some notes will be filtered out. Indeed, Page's phrase learning task consisted of learning very simple nursery rhymes, which were presented isochronously with each note corresponding to a beat at a particular metrical level.

- Another benefit of enabling a SONNET module to become event-driven is that such modules can be embedded within a SONNET cascade when forming a hierarchy (see Section 6.4). Classifications will occur at different rates on different levels in a hierarchy. For example, at $\mathcal{F}_1$ events occur relatively frequently. At $\mathcal{F}_2$ events occur only when patterns at $\mathcal{F}_1$ are recognized and subsequently chunked-out. Consequently, events at $\mathcal{F}_2$ occur at a slower rate than events at $\mathcal{F}_1$. This trend continues as additional modules are appended. Therefore, with the use a time-driven network, it would be extremely difficult to set the operational parameters of the network such that regularities at these higher levels can be discovered. However, this problem can be avoided by operating the higher level SONNET modules in event-driven mode. Since event-driven networks are invariant of presentation rates, and indeed their position within a hierarchy, the number of items at the event level that constitute a learned pattern is of no concern. This enables the same operational parameters to be used at every level of the hierarchy.

Nigrin [114, Chapter 5] proposes making SONNET partially time-driven and partially event-driven by altering the way in which the attentional signal operates. Rather than the attentional signal being binary as described thus far, the signal could be made continuous. For example, by allowing the signal to reach high levels when an event occurs (say unity) and then letting the signal gradually degrade over time to low levels (but never to zero), the network would effectively be operating partially in event-driven mode and partially in time driven mode. For more information on the use of attentional signals with SONNET see Nigrin [114, Section 5.3].

6.4 Cascading SONNET modules to form a hierarchy

This section will describe how separate SONNET modules can be cascaded to form a hierarchy (see [114, Chapter 5] and [118, Chapter 7]). Hierarchies allow categories to form for more abstract concepts, such as phrase sequences. Such a configuration of SONNET modules is hugely advantageous, particularly because regularities at the phrase level can be used to unambiguously generate specific, veridical expectations, leading to accurate melody recall (see Section 7.2). Furthermore, in addition to recognizing singular phrases, the provision of a hierarchy enables information regarding phrase sequences to be utilized to provide a global context, thus significantly enhancing melody recognition (see Chapters 11 and 13). This section will show how *two* SONNET modules can be joined to form a hierarchy. To distinguish the fields of each module composing the hierarchy, two additional alphabetic indices are utilized. $\mathcal{F}_1^a$ and $\mathcal{F}_2^a$ refer to the fields used to process event occurrences, whereas $\mathcal{F}_1^b$ and $\mathcal{F}_2^b$ refer to the fields used to process sequences of pattern classifications.

Thus far, each field of a SONNET module performs a unique task. $\mathcal{F}_1$ transforms a temporal pattern of input items into a non-ambiguous spatial pattern of activity, whereas $\mathcal{F}_2$ classifies these spatial activity patterns. However, when SONNET modules are cascaded, $\mathcal{F}_2^a$ (classification layer) needs to be amalgamated with $\mathcal{F}_1^b$ (input layer) to form a two layer homologous field at the point of contact between the connecting SONNET modules. In future we refer to this amalgam as simply $\mathcal{F}_2$ (see $\mathcal{F}_2$ in Figure 6.8). So, spatial patterns at $\mathcal{F}_1$ are classified by F_2. These classifications, and the order in which they occur are then classified by $\mathcal{F}_3$. Thus, $\mathcal{F}_1$ represents familiar items, $\mathcal{F}_2$ represents familiar sequences of items and $\mathcal{F}_3$ represents familiar sequences of sequences. Thus, the use of a hierarchy allows sequences longer that the TMS to be learned by a single cell assembly at $\mathcal{F}_3$. Note that the first and last fields of a hierarchy do not need this two-layer structure as they communicate directly with their external environment, in the same way as a standard SONNET module does. The manner by which SONNET modules are joined is straightforward. In fact, the ease with which SONNET modules can form hierarchies is one advantage SONNET has over some other forms of neural network hierarchies.

The propagation of classifications at the classification layer to the input layer within $\mathcal{F}_2$ is mediated by an additional cell called a latch cell, which is denoted by l_i. Classification cells at $\mathcal{F}_2$ operate in the same way as before. When a cell assembly at $\mathcal{F}_2$ recognizes its pattern across $\mathcal{F}_1$ and reaches its classification threshold, K_{13}, for a time period of K_{14}, it chunks-out its pattern and resets its classification cells in the normal way. At this point, in addition to sending direct chunk-out signals to $\mathcal{F}_1$, a signal is also sent to the latch cell. In response, l_i will activate for a short time period of K_{latch} and trigger the attentional pulse at the phrase sequence module. This in turn activates s_i at the $\mathcal{F}_2$ input layer, indicating to $\mathcal{F}_3$ the occurrence of a classification at $\mathcal{F}_2$.

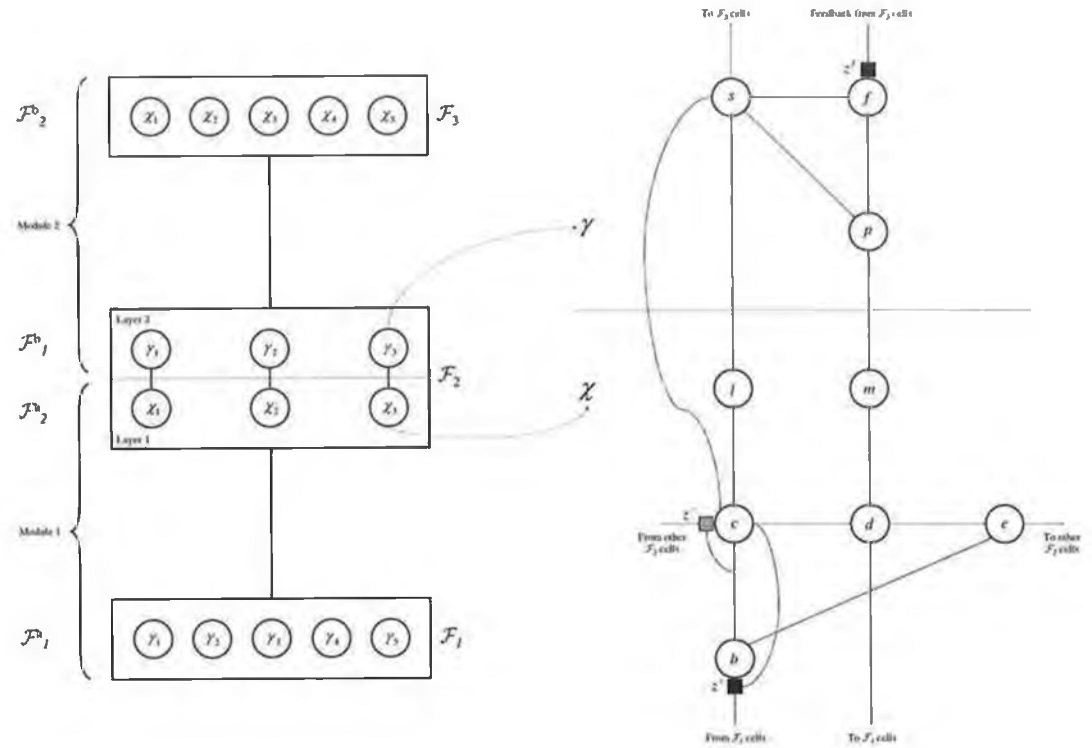

$\mathcal{F}^b_2$
χ_1 χ_2 χ_3 χ_4 χ_5
$\mathcal{F}_3$
Module 2
$\mathcal{F}^b_1$
Layer 2
γ_1 γ_2 γ_3
$\mathcal{F}_2$
$\mathcal{F}^a_2$
χ_1 χ_2 χ_3
Layer 1
Module 1
$\mathcal{F}^a_1$
γ_1 γ_2 γ_3 γ_4 γ_5
$\mathcal{F}_1$
γ
χ
To $\mathcal{F}_3$ cells
Feedback from $\mathcal{F}_3$ cells
z^f
s
f
p
l
m
z^-
From other $\mathcal{F}_2$ cells
c
d
e
To other $\mathcal{F}_2$ cells
b
z^+
From $\mathcal{F}_1$ cells
To $\mathcal{F}_1$ cells

Figure 6.8: Cascading SONNET modules to form a hierarchy. Note that the cells p and m are used to percolate expectancies down the hierarchy and will be discussed at greater length in Section 7.2.

The input layer at $\mathcal{F}_2$ records these classifications and the order in which they occurred as usual, with a decreasing activity pattern. Note that it makes no difference whether s_i is activated by an external environmental stimulus or internally due to a classification; its behaviour is identical.

In addition to recording the order of classifications, s_i cells inhibit c_i cells for as long as they remain active. This ensures that c_i does not respond to another occurrence of its pattern while s_i is active. This would result in $\mathcal{F}_2$ failing to correctly communicate a phrase repetition to $\mathcal{F}_3$. However, this leads to the important question of which cell assembly should respond to a phrase repetition if c_i is inhibited in this manner. The answer is to arrange $\mathcal{F}_2$ into siblings and groups in the same fashion as $\mathcal{F}_1$. This will enable phrase repetitions to be represented. However, although phrase repetition can be represented at $\mathcal{F}_2$ using this structure, the manner by which they can be correctly processed has, thus far, undergone little investigation. Nonetheless, new work in Section 8.2 introduces a mechanism that enables phrase repetitions to be handled correctly.

6.5 Summary

This chapter presented some of the more advanced topics in SONNET research; repetition handling, the use of attention and the formation of hierarchies. In addition to Chapter 5, the groundwork necessary to discuss recent applications of SONNET to melody processing has been laid. Moreover, the issues discussed in this chapter provide the necessary background information for new research on SONNET and the CBR of melodies, which is discussed in Part III of this thesis.

Chapter 7

Previous Applications of SONNET to Melody Processing

7.1 Introduction

Chapters 5 and 6 presented the SONNET neural network (both fundamental and advanced topics), which was designed by Nigrin to process temporal patterns. This chapter presents further advances in SONNET research, which corroborate the strength of SONNET for processing temporal patterns, and in particular, musical sequences. Research by Page [118, 119] shows how SONNET hierarchies can utilize feedback in order to provide plausible, veridical expectancies, which enable the recall of entire melodies. Further research by Roberts [133, 134, 135] demonstrates how SONNET can be modified to enable it to segment rhythmic patterns in a manner consistent with certain aspects of rhythm perception.

7.2 Page's work on generating expectancies

Page trained SONNET with an input set consisting of twelve simplified nursery rhyme melodies. Melodies were represented as sequences of diatonic scale notes (of which there are seven), with each note lasting the duration of an attentional pulse, which was fired in response to a beat at the appropriate metrical level. The primary goal of Page's work, once learning had already taken place, was to utilize the top-down feedback weights to produce plausible, veridical expectations in response to incomplete melodic phrases presented at $\mathcal{F}_1$. This enabled the recall of entire melodies.

7.2.1 Generating expectancies using top-down feedback

The expectation, p_{mj}, for an event occurrence is defined in terms of the feedback signals received at f_{mj}[1], with greater top-down signals yielding greater expectations. Provided s_{mj} (at γ_{mj})[2] is not already active (as events that have already occurred cannot be expected), the expectation at an input assembly is calculated as follows:

$$p_{mj} = \begin{cases} f_{mj} & \text{if } s_{mj} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

$$f_{mj} = \max_{i \in P}(d_i z^f_{ikj}) \tag{7.2}$$

Note that f_{mj} is calculated using the maximum top-down signal rather than the sum of the top-down signals (see Equation 5.28). Consquently, only the feedback signal from the cell assembly at $\mathcal{F}_2$ which best represents the partial input pattern at $\mathcal{F}_1$ is utilized.

During Page's preliminary expectancy work, various interference effects were discovered that prohibited accurate expectancy generation. To overcome these interference effects, Page introduced a modification that placed restrictions on the set of $\mathcal{F}_2$ cell assemblies, labelled P, which are permitted to generate top-down priming signals at $\mathcal{F}_1$. To aid in the construction of P, the following calculations were introduced:

$$W_i^+ = \sum_{k,j \in T_i} W_{kji}^+ \tag{7.3}$$

$$W_{kji}^+ = \begin{cases} 1 & \text{if } K_4 S_{kji} > S_{0i} \text{ and } S_{0i} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.4}$$

$$W_i^\times = \max\left(\prod_{k,j \in T_i} W_{kji}^\times, K_8\right) \tag{7.5}$$

$$W_{kji}^\times = \begin{cases} K_1 + K_2 & \text{if } K_4 S_{kji} > S_{0i} \text{ and } S_{0i} > 0 \\ K_1 & \text{otherwise} \end{cases} \tag{7.6}$$

These calculations are analogous to those used to compute a cell assembly's gated input (I_i^+) and a cell assembly's confidence ($I_i^\times$). However, their calculation is based on the secondary weights w_{kji}^+ rather than the excitatory weights z_{kji}^+. Using these quantities, the following conditions specify which cell assemblies may become members of the set P and provide expectancy signals to $\mathcal{F}_1$ input assemblies:

[1] Page used the label t for identifying the cell at each $\mathcal{F}_1$ input assembly that computes the feedback signal from $\mathcal{F}_2$. However, in Nigrin's original SONNET specification, the label f was used. This thesis will remain consistent with Nigrin's original specification.

[2] Remember, in Chapter 5, the term γ_j was introduced to collectively refer to each set of cells that process particular event occurrences. Furthermore, since $\mathcal{F}_1$ is now organized into sibling groups (see Section 6.2), each input assembly is labelled accordingly — γ_{mj}.

χ_i **is committed.** By ensuring that a cell assembly is committed, before it is allowed to become a member of the set P, interference effects from cells assemblies that have yet to fully learn a pattern will be eliminated.

χ_i **is on-target.** χ_i is considered as on-target if the pattern it encodes is entirely consistent with the current input at $\mathcal{F}_1$. For example, suppose the sequences {1, 2, 3, 4} and {1, 3, 2, 4} have been learned by χ_{1234} and χ_{1324} respectively. Furthermore, suppose the sequence {1, 2, 3} has been presented at $\mathcal{F}_1$. In this case, χ_{1234} is consistent with the pattern at $\mathcal{F}_1$ because the beginning of its pattern is active. Therefore, χ_{1234} is considered as on-target and will become an element of the set P. However, χ_{1324} is inconsistent with the pattern at $\mathcal{F}_1$ and is not considered as on-target. Consequently, χ_{1324} should not become an element of P. This will prevent any interference effect from χ_{1324}. The quantity $W_i^\times$ can be utilized to discover if χ_i is on-target or not. Since $W_i^\times$ represents the largest possible $I_i^\times$ at χ_i at a particular point in time, a value of $I_i^\times \approx W_i^\times$ signifies that a cell assembly is on-target. That is, a cell assembly is on-target if:

$$I_i^\times > W_i^\times K_{on-target} \tag{7.7}$$

$K_{on-target}$ should be set within the following range $[0.7 \leq K_{on-target} \leq 0.9]$.

χ_i **is warranted.** Another form of interference concerns the problem of unwarranted expectations. Consider the cell assemblies χ_{1234} and χ_{345} that have learned the sequences {1, 2, 3, 4} and {3, 4, 5} respectively. Suppose that both cell assemblies are responding to the input pattern {1, 2, 3, 4} at $\mathcal{F}_1$. In this circumstance, both cell assemblies are committed and are on-target; however, an expectation for the occurrence of a 5 would be yielded. This is inconsistent with the fact that χ_{1234}'s complete pattern has been presented, indicating the closure of a phrase. Consequently, the expectation of a 5 is considered unwarranted. To avoid interference from such cell assemblies the quantity W_i^+ may be utilized. Therefore, the following condition must also be met before a cell assembly may become a member of the set P:

$$W_i^+ > W_{max}^+ K_{warranted} \tag{7.8}$$

where W_{max}^+ is the largest W^+ value found at a committed, on-target cell assembly. Furthermore, $K_{warranted}$ should be set within the following range $[0.8 \leq K_{warranted} \leq 1.0]$

If no input assembly at $\mathcal{F}_1$ is active, then all committed $\mathcal{F}_2$ cell assemblies will be included in the set P. These restrictions on membership of P removed all of the interference effects described and provided a dramatic improvement in the expectations that were generated in Page's empirical results.

Note that Page originally used different parameter names for $K_{on-target}$ and $K_{warranted}$ (K_{15} and K_{16} respectively). However, these parameter names had already been reserved for use by Nigrin. Consequently, to avoid a name clash, these new parameter names are introduced here.

7.2.2 Hierarchical expectation and complete melody recall

Section 6.4 showed how SONNET modules can be organized in a hierarchical fashion and how classifications of patterns occurring at $\mathcal{F}_1$ can be communicated to higher levels. This section shows how Page [118, Chapter 7] utilized this hierarchical organization to enable SONNET to recall entire melodies. However, instead of propagating *classifications* up the hierarchy, *expectations* are propagated down the hierarchy.

Page showed that a larger context was sometimes needed to produce correct expectations at phrase boundaries due to the problem of smaller phrases dominating larger phrases with regard to produced expectancies. This context can be provided using a SONNET hierarchy. Consequently, Page introduced the means by which SONNET hierarchies can be utilized *"in order to provide veridical memory of extended, multi-phrase sequences"* [118, pg. 241] (note that Chapters 10 and 12 illustrate the additional need to arrange SONNET modules in a parallel fashion). The manner by which phrase expectations can be sent down through a hierarchy is now explained.

Using an inter-module priming field for communicating phrase expectations

An abstract SONNET module (for example, $\mathcal{F}_1^b$ and $\mathcal{F}_2^b$) can communicate its phrase expectations to a lower level SONNET module (for example, $\mathcal{F}_1^a$ and $\mathcal{F}_2^a$) using an ***inter-module priming field***, consisting of priming cells labelled m_i, located between $\mathcal{F}_2^a$ and $\mathcal{F}_1^b$. There is a one-to-one correspondence between each priming cell m_i and its associated input assembly (γ_i at $\mathcal{F}_1^b$) and cell assembly (χ_i at $\mathcal{F}_2^a$) (see Figure 6.8). This priming field is analogous to the field of latch cells used to propagate classifications from the bottom upwards. Instead, however, phrase expectations are propagated from the top downwards. Each priming cell's activation level is calculated as follows:

$$m_i = \begin{cases} 1 & \text{if } p_i > 0.85 p_{max} \text{ and } p_{max} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

where p_i refers to the expectation at γ_i in $\mathcal{F}_1^b$ and p_{max} is the maximum expectation found at $\mathcal{F}_1^b$. The constant 0.85 ensures that only genuine ambiguity is propagated, whilst filtering out interference from expectations for phrases that are not expected next, but perhaps at some time in the future. In this way, $\mathcal{F}_3$ can communicate phrase expectations to $\mathcal{F}_2$ (remember, in the case of a hierarchy, $\mathcal{F}_2$ is considered an amalgam of $\mathcal{F}_2^a$ and $\mathcal{F}_1^b$). The activation level of the priming cells can now be utilized to influence processing at $\mathcal{F}_1^a$ and $\mathcal{F}_2^a$.

Percolating phrase expectancies down a hierarchy

In order to allow the generation of veridical expectations, a phrase expectation can be percolated down a hierarchy to $\mathcal{F}_1$ via the feedback cells (d_i) at each $\mathcal{F}_2$ cell assembly. To achieve this, d_i is now calculated as follows:

$$d_i = \min(c_i + m_i K_{prime}, 1.0) \tag{7.10}$$

where K_{prime} is set to 0.3.

Facilitating the chunking-out of expected phrases

Phrases that are expected can be chunked-out immediately, once they receive their full input. To achieve this:

1. A primed cell assembly is given a competitive advantage by modifying the excitatory cell b_i in the following way:
$$b_i \begin{cases} 2I_i^+ I_i^\times & if\ m_i = 1 \\ I_i^+ I_i^\times & otherwise \end{cases} \tag{7.11}$$
2. Cell assemblies that are primed ($m_i = 1$) and respond to their complete pattern ($M_i \geq K_M$) are not subject to any chunking-out delay.

In this way, an expectation can enable a cell assembly that represents the expected phrase to be instantly processed and chunked-out out in response to its complete pattern at $\mathcal{F}_1$. This facilitates the generation of expectations for the subsequent phrase. Thus, a network hierarchy provides the context necessary to avoid the domination of small patterns over large pattern with respect to generating expectancies. In fact, complete expectancies for the next phrase at $\mathcal{F}_1$ can be generated, even before any note of that phrase has actually been presented. This is what is referred to as melody recall within the SONNET framework.

7.3 Roberts' work on interpreting rhythmic structures

This section describes research by Roberts [133] that shows how SONNET can be configured to respond directly to the IOI between consecutive event onsets using a time-driven network configuration. Therefore, the rhythmical features of an input sequence can be processed. Although Roberts does show how isochronous sequences of quantized IOIs can be processed using an event-driven network, no new concepts or modifications are required to achieve this task, over what has already been presented in this thesis so far. Furthermore, for processing time intervals, Roberts has shown that the methods and modifications described next, better model certain aspects of rhythm perception over the use of quantized IOIs.

7.3.1 Representing IOIs at $\mathcal{F}_1$ using cell activation ratios

Using a time-driven SONNET configuration to process IOIs requires the use of only a single sibling group, which represents event onsets. Since only a single sibling group is utilized, the indexing scheme can be simplified from that presented in Section 6.2.5. In this section, each input assembly is denoted by s_j, and each excitatory weight from $\mathcal{F}_1$ to each $\mathcal{F}_2$ cell assembly will be denoted by z_{ki}^+, where $k = 1$ references the strongest link and $k = N_l$ references the weakest link.

By allowing the network cells and activations to continually change, the relative activation levels between input cells, which respond to consecutive event onsets, will represent each IOI (refer back to Section 6.3.1). Although Equation 5.1 may be used to achieve this behaviour, Roberts introduced a simplified version of this to minimize processing time. Therefore, input cells are governed by the following equation:

$$s_j(t + \Delta t) = \omega^{\frac{1}{24}} s_j(t) \tag{7.12}$$

where $\Delta t = T_{span}/24$ and T_{span} represents a melody's tactus-span. In effect, the time interval between consecutive event onsets, in terms of the tactus-span, is represented. Since a single sibling group is used, the bottom-up input to each $\mathcal{F}_2$ cell assembly is insensitive to the spatial arrangement of $\mathcal{F}_1$. Consequently, it is more difficult to distinguish different but similar activity traces across $\mathcal{F}_1$. To help with this, $\omega = 3$ in order to provide a greater degree of contrast between similar but different patterns.

7.3.2 Using a sliding window overload scheme

Section 5.3.1 discussed various possibilities for limiting the depth and capacity of STM at $\mathcal{F}_1$. Among these possibilities was the use of a sliding window storage scheme, so called because it behaves like a fixed size window (specified by K_{12}) that continually shifts across an input sequence, one event at a time. Roberts chose this mechanism in preference to others because it enables time to have a greater influence over processing, which is necessary when processing sequences of time intervals. In Roberts experiments, every input cell was reset to zero once it had been active for at most six tactus-spans ($K_{12} = \omega^6$).

Using the techniques described so far, IOIs can be represented at $\mathcal{F}_1$ by the relative activations between input cells representing consecutive event onsets. Furthermore, the overload mechanism enables time to have a greater influence on processing. However, patterns that are represented by activity traces alone are much more difficult to process than those whose spatial dimensions are available for utilizing. Although increasing ω at $\mathcal{F}_1$ helps the situation somewhat, a number of modifications at $\mathcal{F}_2$ are still required. These modifications, which were introduced by Roberts [133, Chapters 5 and 6], are briefly discussed next.

7.3.3 Modifications to the set T_i

Roberts [133, Section 5.8.2] points out two problems with the current secondary weight formulation (Equation 5.16) when it is utilized to process IOIs using a single sibling group:

1. Using Nigrin's original formulation, T_i identifies the specific input items that cause c_i's activity to become high. However, when $\mathcal{F}_1$ uses a single sibling group, which represents the same type of item (that is, event onsets), this mechanism does not work.

2. The maximum $|T_i|$ was restricted by using the term $K_4 S_{ji} > S_{0i}$. This compared normalized input cell activations with one another. In this way K_4 specifies the maximum length of the set T_i. However, when timing information is being represented at $\mathcal{F}_1$ in the form of IOIs, this restricts the duration of each rhythmic pattern, rather than the specific number of event onsets that may comprise a pattern.

To address these issues, Roberts introduced the following secondary weight formulation:

$$\frac{\partial}{\partial t} w_{ji}^{+} = \epsilon_1 r_{ji} \min\left(c_i^2, K_w\right) \left[-L_i w_{ji} + I_{ji}^{\times} c_i\right] \tag{7.13}$$

where ϵ_1, r_{ji} and L_i are the same values used in the excitatory weight calculations. K_w is a small constant, which ensures that a cell assembly can improve its pattern representation before the set T_i changes. This formulation measures the frequency at which a specific pattern of relative cell activities occurs at $\mathcal{F}_1$, rather than measuring the frequency at which individual input items are presented at $\mathcal{F}_1$. The term $I_{ji}^{\times}$ was incorporated into the w_{ji}^{+} formulation to achieve this behaviour.

Since c_i is generally high when it responds to its pattern, each link that forms χ_i's pattern will have a high confidence value ($I_{ji}^{\times} \approx 1$). Consequently, links that have high confidences simultaneously with large c_i values will evolve to have large secondary weights; therefore these links become members of T_i. Conversely, low confidence values when c_i is high will reduce the secondary weights on these links, which will not become members of T_i. In this way, the secondary weight formulation is insensitive to the spatial layout of $\mathcal{F}_1$, which addresses the first issue. (Note that, Nigrin [114, Section 3.16.2] introduces a similar mechanism in an attempt to use a separate vigilance at each bottom-up link).

The second problem was addressed by removing the term $K_4 S_{kji} > S_{0i}$ from the the secondary weight learning equation. Instead, the number of bottom-up links to each cell assembly (specified by K_l) is used to restrict the maximum number of grouped onsets of a rhythmic pattern. In Roberts' simulations, K_l was set to 5 in order to encode a pattern consisting of four IOIs, which is consistent with studies in human rhythm perception.

To avoid learning single onsets, which do not encode any IOI information, Roberts points out that having the Z_{ji} calculation analogous to the S_{ji} calculation should be avoided. Instead, z_{ji}^{+} is normalized over the length of the entire weight vector (z_i^{tot}) to bias the network against encoding patterns consisting of a single onset. In fact, there is practically no difference between the two normalization strategies once χ_i has formed a good pattern representation. Consequently, no restriction was placed on the minimum number of grouped onsets allowed.

7.3.4 Modifying the chunking-out mechanism

Roberts [133, Section 5.7] identified a problem that occurs with the chunking-out mechanism when $\mathcal{F}_1$ represents IOI information using cell activation ratios. When a pattern is chunked-out, it is highly probable that the last onset of the pattern may encode the beginning of the subsequent pattern. Therefore, IOI information may be lost from $\mathcal{F}_1$. This problem can be easily avoided by ensuring that the input cell that represents the last onset of every pattern is not reset during the chunking-out process.

7.3.5 Initializing z_{ji}^{+} to bias the network towards grouping by temporal proximity

The excitatory weights, z_{ji}^{+}, are initialized within the range $[K_{zmin}, K_{zmax}]$. Setting K_{zmin} and K_{zmax} over a broad range would bias SONNET towards encoding rhythmic patterns consisting of relatively long IOIs. This is because relative excitatory weight values would more easily match patterns presented at $\mathcal{F}_1$ that consist of relatively long IOIs. Consequently, $\mathcal{F}_2$ cell assemblies would initially gain higher activations in response to these patterns.

However, as Chapter 2 pointed out, rhythmic events are more likely to be grouped if they occur relatively close together in time; this is known as grouping by temporal proximity. Therefore, in order to bias the network towards grouping by temporal proximity, K_{zmin} and K_{zmax} should be initialized over a narrow range. Roberts [133, Appendix D] used values of $K_{zmin} = 0.06$ and $K_{zmax} = 0.08$ to achieve this.

7.3.6 Controlling the influence of IOIs on $\mathcal{F}_2$ activations

Due to the operation of SONNET in time-driven mode, relatively long IOIs may act as phrase boundary indicators that separate different IOI patterns. This is consistent with the gap principle of rhythm perception. However, the gap effect exhibited by SONNET can be too extreme for modelling rhythm perception. This is because long IOIs enable c_i to have a large activity for the entire duration of the IOI. In turn, the large c_i activity causes the pattern at χ_i to be learned very quickly. However, as Roberts [133, Section 5.8.4] points out, studies into rhythm perception have shown that long IOIs do not have such a pronounced effect on learning as time passes. This is because long IOIs (greater that 2 seconds) are difficult to perceive. To curtail this effect, Roberts introduced a *time dependence* parameter, τ_i, to limit a cell assembly's activation growth:

$$\frac{\partial}{\partial t}c_i = -Ac_i + \left(\frac{B - c_i}{D_i}\right)\left(\frac{v_1 b_i + e_i}{\tau_i}\right) - \frac{Q_i}{D_i}v_2 Q_i \tag{7.14}$$

where,

$$\tau_i = (t_i^{'})^{(1-I_i^{+max})} \tag{7.15}$$

$$t_i^{'} = \begin{cases} t_i & \text{if } t_i > 0.3 \\ 0.3 & \text{otherwise} \end{cases} \tag{7.16}$$

where $I_i^{+max} = b_i / I_i^{\times}$ whenever $M_i > K_M$. τ_i divides excitatory input, similarly to the dilution parameter D_i, but it is time-based rather that $|T_i|$-based. Since τ_i increases with absolute time, a cell assembly's competitive strength does not continually increase, thus curtailing excessive learning at χ_i due to relatively long IOIs.

7.3.7 Additive confidence formulation

Section 5.3.2 described the confidence value $I_i^\times$, which compares the normalized weight vector against the normalized input vector, thus measuring the confidence with which a cell assembly matches the input pattern at $\mathcal{F}_1$. However, a number of problems exist with this formulation when processing IOIs using a time-driven network. Firstly, due to K_8, $I_i^\times$ is unable to distinguish between arbitrary levels of mismatch between normalized excitatory weights and normalized inputs. Secondly, $I_i^\times$ tends to discriminate against small patterns, which hindered Roberts' modelling of rhythm perception. To address these issues, Roberts [133, Section 6.3] introduced the following confidence formulation:

$$I_i^\times = 1 + \frac{K_2}{|T_i|}\left(\sum_{j \in T_i} I_{ji}^\times\right)^2 \tag{7.17}$$

$$I_{ji}^\times = \min\left(\frac{Z_{ji}}{S_{ji}}, \frac{S_{ji}}{Z_{ji}}\right) \tag{7.18}$$

7.3.8 Inhibitory weights

Nigrin's original inhibitory weight-learning mechanism (see Equation 5.3.6) evolved such that:

- If the patterns coded by χ_j and χ_i had any input items in common, the inhibitory weights between these cell assemblies would equilibrate to unity ($z_{ji}^- = z_{ij}^- = 1$).
- If the patterns coded by the *committed* cell assemblies χ_i and χ_j have *no* input items in common, the inhibitory weights between these cell assemblies would be set to and frozen at zero ($z_{ji}^- = z_{ij}^- = 0$). These cell assemblies can classify their respective patterns simultaneously.

However, when using a single sibling group representing event onsets, all items are common across all patterns learned at $\mathcal{F}_2$. Consequently, similarity cannot be defined in terms of common input items because every pattern will have an input item in common; that of an event onset. Therefore, Nigrin's inhibitory weight freezing mechanism cannot be utilized when a single sibling group is used at $\mathcal{F}_1$. Consequently, disjoint patterns cannot be classified simultaneously. Since the ability to classify disjoint patterns simultaneously enhances context sensitive recognition, Roberts introduced a new inhibitory weight formulation, in which similarity is defined in terms of pattern overlap rather than common input items. This is achieved by comparing the confidence values between competing cell assemblies:

$$\frac{\partial}{\partial t} z_{ij}^- = \epsilon_1 \epsilon_3 c_i^2 \min\left[(I_i^{+max})^{-5}, K_{inh}\right]\left[-z_{ji}^- + \left(\frac{I_j^\times}{\rho_i}\right)\right] \tag{7.19}$$

This formulation is stable because it depends on the simultaneity of bottom-up inputs. If $I_j^\times$ and $I_i^\times$ are simultaneously large, then z_{ji}^- tends to unity; thus the cell assemblies compete. The connection becomes stronger at the cell assembly with the lower confidence, and weaker at the cell assembly with the higher confidence. In this way, over time, two cell assemblies can become non-competitive if the patterns they are encoding do not overlap. Furthermore, since the weights equilibrate to values that represent the level of overlap between these patterns, rather than the binary values of 1 or 0 that characterize Nigrin's formulation, the contrast between bottom-up inputs is emphasized.

The term $\min\left[(I_i^{+max})^{-5}, K_{inh}\right]$ allows weights to become effectively frozen when cell assemblies commit to their pattern. However, before then, the weights can evolve quickly. In effect, this learning-rate regulator provides *"selectivity when learning and co-activation when classifying"* [133, pg. 169]. Finally, the vigilance parameter, ρ_i, is given by:

$$\rho_i = \rho I_i^{\times} \tag{7.20}$$

Note that ρ does not need to be an exponent (as in Nigrin's formulation) because $I_i^{\times}$ increases linearly with $|T_i|$.

7.3.9 Contrast enhancement

Due to the introduction of the new confidence formulation and the new inhibitory learning formulation, Roberts found that, during preliminary network simulations using his rhythm test suite, the network failed to adequately differentiate between different input patterns across $\mathcal{F}_1$. Consequently, slow learning resulted. To address this issue, Roberts introduced simple modifications that enhanced the contrast between cell assembly activation levels.

Faster-than-linear signals

Since cell assemblies compete via the inhibitory cells e_i, the contrast between cell assemblies can be increased by altering the activation equation for inhibitory cells (d_i^3 rather than d_i^2):

$$e_i = b_i d_i^3 \tag{7.21}$$

To increase the contrast between bottom-up links, the learning equations at these links were also modified sightly by introducing the term c_i^2 rather than c_i:

$$\frac{\partial}{\partial t} z_{ji}^{+} = \epsilon_1 r_{ji} c_i^2 \left[-L_i z_{ji}^{+} + S_{ji} c_i\right] \tag{7.22}$$

$$\frac{\partial}{\partial t} w_{ji}^{+} = \epsilon_1 r_{ji} \min\left(c_i^2, K_w\right)\left[-L_i w_{ji} + I_{ji}^{\times} c_i\right] \tag{7.23}$$

These changes enhanced the stability of learning because the weights were prevented from modulating in response to significantly different patterns. Note that the contrast enhancement at the secondary weights are provided by the term $\min\left(c_i^2, K_w\right)$, which has already been specified in Section 7.3.3.

Additionally, I_i^{max} was also governed by c_i^2 when $I_i^{max} \geq I_i^{\times}$:

$$\frac{\partial}{\partial t} I_i^{max} = \epsilon_1 r_{0i} c_i^2 \left[-I_i^{max} + I_i^{\times}\right] \tag{7.24}$$

This modification was required because the difference between I_i^{max} and $I_i^{\times}$ was now smaller due to the new confidence formulation.

Modifying an $\mathcal{F}_2$ cell assembly's excitatory input

Roberts [133, Section 6.5.2] found, due to the additive $I_i^\times$ formulation (see Section 7.3.7), that a cell assembly, χ_i, with a good pattern representation could achieve high activations in response to patterns at $\mathcal{F}_1$ that differ significantly from χ_i's encoded pattern. Consequently, χ_i could inhibit competing cell assemblies, which hindered the learning of other patterns.

To overcome this problem, Roberts noted that (due to the modification discussed in Section 5.5.4), that D_i can increase more easily than it can decrease. Consequently, $I_i^\times < D_i$ when it is responding to patterns other than that which it encodes. Therefore, the ratio $\frac{I_i^\times}{D_i}$ can be used to provide a greater contrast between $\mathcal{F}_2$ cell assemblies:

$$\left(\frac{B - c_i}{D_i}\right)\left(\frac{v_i b_i + e_i}{\tau_i}\right)\left(\frac{I_i^\times}{D_i}\right)^{3I_i^{+max}} \tag{7.25}$$

Modifications to the link interactions

Thus far, the application of multiple interacting bottom-up links enables a cell assembly χ_i to respond to the $|T_i|$ most recent event onsets, provided χ_i has not already recognized its pattern. However, when the $|T_i|$ most recent onsets do not represent the pattern encoded by χ_i, it is useful for χ_i to respond to fewer than the $|T_i|$ most recent onsets. This would enable χ_i to recognize when the beginning of its pattern is present at $\mathcal{F}_1$. This helps thwart the formation of overlapping patterns by other cell assemblies, which could potentially lead to inefficient sequence segmentations. The modification presented here, enables χ_i to respond to fewer than the $|T_i|$ most recent event onsets.

To achieve this, all active URCs on links to χ_i will be reset when $I_{ji}^\times < K_I$. A reset URC will remain inactive for as long as its associated input cell is active. Furthermore, if the URC is associated with the strongest link, the reset signal is propagated to all the LRCs within its link cluster, thus shutting-off the entire link cluster. This reset mechanism should not be invoked when χ_i is responding to a single event onset. This is because it takes at least two event onsets to define an IOI when cell activation ratios are used to represent IOIs. For more information on this mechanism, see Roberts [133, Section 6.6.1].

Boosting c_i at on-target cell assemblies

The term $W_i^\times$ was introduced by Page [118, Section 6.6.1] and employed by Roberts to enable activation levels to be boosted at on-target cell assemblies. This reduces the number of overlapping patterns formed, thus enhancing pattern recognition. However, since Roberts modified the confidence formulation (see Section 7.3.7), this needed to be reflected in the calculation of $W_i^\times$:

$$W_i = \begin{cases} W_i^\times & \text{if } I_i^{+max} > K_{embed} \\ D_i & \text{otherwise} \end{cases} \tag{7.26}$$

$$W_i^\times = 1 + \frac{K_2}{|T_i|}\left(\sum_{j\in T_i} W_{ji}^\times\right)^2 \tag{7.27}$$

$$W_{ji}^\times = \begin{cases} 1 & \text{if } S_{ji} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7.28}$$

where the term W_i was introduced into the c_i activation equation as follows:

$$\left(\frac{B - c_i}{W_i}\right)\left(\frac{v_i b_i + e_i}{\tau_i}\right)\left(\frac{I_i^\times}{D_i}\right)^{3I_i^{+max}} \tag{7.29}$$

Consequently, for on-target cell assemblies, the dilution parameter is reduced causing a boost to c_i for as long as these cell assemblies remain on-target.

7.4 Summary

This chapter presented previous applications of SONNET to melody processing. In particular, work by Page showed how phrase expectancies can be produced, thereby enabling the complete recall of entire melodies. Further work by Roberts was presented, which showed how SONNET could be utilized to segment rhythmic patterns. Even taking these modifications into account, a number of problems still exist with the SONNET formulation. These problems will be discussed in the next chapter, which introduces solutions to these problems.

Part III

New SONNET and CBR Research

Chapter 8

Improvements to SONNET

8.1 Introduction

Chapters 5, 6, and 7 presented all of the developments made in SONNET research, from its inception by Nigrin, to its applications in melody recall by Page and the modelling of rhythm perception by Roberts. However, with the advances made in this dissertation regarding architecture, representation and application to melody retrieval, a number of limitations and problems with the incumbent formulations were identified. This chapter discusses these problems and specifies improvements that address these issues. In particular:

- No previously described SONNET formulation can process phrase repetitions adequately.
- No previous multiplicative $I_i^{\times}$ formulation adequately measures arbitrary levels of mismatch between STM and LTM.

Additionally, to show that these modifications operate as intended, we reproduced previous SONNET experiments by Nigrin [114, Chapter 4], Page [118, Chapter 7] and Roberts [133, Chapter 6].

8.2 Processing repeated phrases at $\mathcal{F}_2$

Section 6.2 describes how patterns containing *event* repetitions can be represented at $\mathcal{F}_1$ and subsequently recognized at $\mathcal{F}_2$. This involves presynaptic interactions between links that emanate from different input assemblies at $\mathcal{F}_1$ (within the same sibling group) and terminate at the same cell assembly at $\mathcal{F}_2$. This form of competition is called *intra-cell* competition. In contrast, this section introduces a new mechanism modelled on similar principles, which enables $\mathcal{F}_2$ to represent and recognize *phrase* repetitions at $\mathcal{F}_1$. This involves presynaptic interactions between links that emanate from the same input assembly at $\mathcal{F}_1$ and terminate at different $\mathcal{F}_2$ cell assemblies (within the same sibling group). This form of competition is called *inter-cell* competition.

The ability to represent and recognize phrase repetitions is essential for the correct operation of SONNET hierarchies and, in particular, for the new work on SONNET-MAP which is introduced in Chapter 10. Nigrin [114, Chapter 6] introduced the general principles of how this functionality can be achieved. However, no definitive mechanism or implementation was presented. In this section, one theoretical neural circuit, which would enable SONNET to process phrase repetitions, is introduced. Furthermore, it is shown how this circuitry can easily be modelled and implemented with very little additional computational cost.

8.2.1 Using multiple cell assemblies to represent each learned pattern

Suppose that a single SONNET module has learned the phrase {A, B}, which is represented by χ_{AB}. Furthermore, suppose consecutive occurrences of this phrase have been presented to $\mathcal{F}_1$. That is, the sequence {A, B, A, B} is active at $\mathcal{F}_1$. The second occurrence of the phrase {A, B} may be recognized by χ_{AB} only after the first occurrence of the phrase has been classified and chunked-out. In the case of a SONNET hierarchy (see Section 6.4), the situation is more complex. The second occurrence of the phrase {A, B} may only be completely processed when:

1. The first occurrence of the phrase has been classified and chunked-out by χ_{AB}. This will result in the activation of γ_{AB} via its associated latch cell l_{AB}.

2. γ_{AB} has been reset, either by the chunking-out mechanism at $\mathcal{F}_3$ or by an input field overload at the $\mathcal{F}_2$ input layer.

In fact, it is unlikely that the second occurrence of the pattern {A, B} will be propagated to the $\mathcal{F}_2$ input layer at all. This is because the interactions between $\mathcal{F}_2$ and $\mathcal{F}_3$ occur over a slower time scale than the interactions between $\mathcal{F}_1$ and $\mathcal{F}_2$ (see Section 6.3.2). Therefore, by the time γ_{AB} has been reset at $\mathcal{F}_2$, it is very unlikely that the second occurrence of the pattern {A, B} will still be present at $\mathcal{F}_1$, due to the input field overload mechanism.

To overcome this problem, Nigrin [112, Chapter 8] suggested the following organizational and functional arrangement for the provision of a SONNET hierarchy (described in Section 6.4). $\mathcal{F}_2$ should be organized into sibling groups in the same way as $\mathcal{F}_1$. That is, $\mathcal{F}_2$ should consist of N_2^g sibling groups, each containing N_2^s siblings. Each sibling[1] within a sibling group at $\mathcal{F}_2$ will represent the same learned pattern. Furthermore, classification cells within the same sibling group will not compete with each other. The introduction of sibling groups at $\mathcal{F}_2$ requires the indexing scheme of cell assemblies to be altered. The n^{th} cell assembly within the i^{th} sibling group will be referenced by χ_{ni}.

Using this configuration, all cell assemblies within a sibling group will respond *identically* to event occurrences at $\mathcal{F}_1$. Therefore, to ensure that only one cell assembly activates its corresponding input assembly at the $\mathcal{F}_2$ input layer in response to the classification of a single phrase, some form of competition must be implemented. This competition is implemented at the field of latch cells. Only latch cells that originate from within the same $\mathcal{F}_2$ sibling group should compete, where the winning latch cell will communicate the occurrence of a classification to its corresponding input

[1] At $\mathcal{F}_1$, each sibling consists of a single input assembly, γ_{mj}. In contrast, in the case of a hierarchy, each $\mathcal{F}_2$ sibling consists of a cell assembly, χ_{ni}, at the classification layer and its corresponding input assembly, γ_{ni}, at the input layer above.

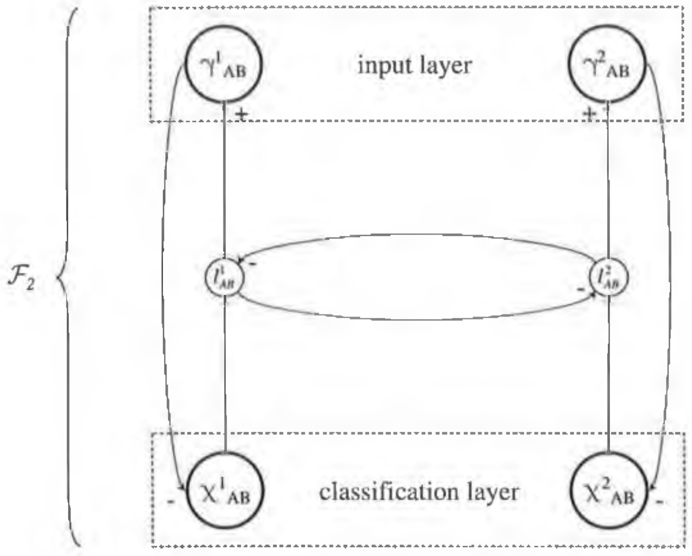


Figure 8.1: The $\mathcal{F}_2$ classification layer and the $\mathcal{F}_2$ input layer for representing repetitions of the phrase {A, B} using Nigrin's approach. χ^1_{AB} and χ^2_{AB} respond identically to phrases at $\mathcal{F}_1$, while the latch cells l^1_{AB} and l^2_{AB} compete such that only one cell assembly will communicate the occurrence of the phrase to the $\mathcal{F}_2$ input layer.

cell. Additionally, while s_{ni} remains active it will inhibit c_{ni}. Consequently, in response to the repetition of a phrase, a different cell assembly within the i^{th} sibling group will activate.

Referring back to the example above, suppose two cell assemblies (χ^1_{AB} and χ^2_{AB}) are utilized to represents the phrase {A, B} (see Figure 8.1). When the the sequence {A, B, A, B} is presented to $\mathcal{F}_1$, both χ^1_{AB} and χ^2_{AB} will activate and respond identically to the first occurrence of their pattern. At some stage, both cell assemblies will pass their classification threshold K_{13} for a time period of K_{14}. This results in the chunking-out of the first occurrence of the pattern {A, B} from $\mathcal{F}_1$. Both cell assemblies will activate their corresponding latch cells l^1_{AB} and l^2_{AB}, which in turn will compete for the right to activate their input cells at γ^1_{AB} and γ^2_{AB} respectively. Suppose in this case, l^1_{AB} wins the competition and activates s^1_{AB}. At this point, both cell assemblies are free to respond to the second occurrence of the phrase at $\mathcal{F}_1$. However in this situation, since c^1_{AB} will be inhibited by s^1_{AB}, the second occurrence of the pattern will be processed by χ^2_{AB}. This results in the activation of l^2_{AB} and ultimately, the activation of γ^2_{AB}. Consequently, both occurrences of the phrase will be recognized at the $\mathcal{F}_2$ classification layer and communicated to the $\mathcal{F}_2$ input layer for processing by cell assemblies at $\mathcal{F}_3$.

8.2.2 Cell assemblies within a sibling group should not respond identically to phrase repetitions

As described thus far, cell assemblies within a sibling group at $\mathcal{F}_2$ respond identically to events at $\mathcal{F}_1$. Therefore, using the example from above, the second occurrence of the phrase {A, B} cannot be processed until the preceding occurrence has been classified and chunked-out. This is true whether or not a hierarchy is used. Consequently, χ^2_{AB} will have a significantly shorter time period during which to respond to the second occurrence of {A, B} than χ^1_{AB} did to respond to the first occurrence of {A, B}. This reduces the chances of χ^2_{AB} classifying and chunking-out its pattern before it gets deleted from $\mathcal{F}_1$, due to the input field overload mechanism. In other words, when cell assemblies within a sibling group at $\mathcal{F}_2$ respond identically to phrases at $\mathcal{F}_1$, the recognition of phrase repetitions can be very sensitive to the parameters that govern the $\mathcal{F}_1$ overload mechanism (K_{12}) and the chunking-out mechanism (K_{13} and K_{14}).

Another problem with this approach relates to the formation of multidimensional phrases where separate categories for pitch sequences and rhythms are fused using the boundary point alignment process introduced in Chapter 10. Unless patterns are recognized as soon as they occur across $\mathcal{F}_1$, this process cannot operate correctly. For example, consider the SONNET-MAP architecture which contains two SONNET modules; one for processing pitch sequences and one for processing rhythms. Suppose the pitch module has learned the following sequence of pitch-intervals {1, 2, 1} and that the rhythm module has learned the following IOI sequence {0.5, 0.5, 1.0}. Furthermore, suppose the following melodic input has been presented to SONNET-MAP:

$$\{\{1, 0.25\}, \{2, 1.0\}, \{1, 0.5\}, \{2, 0.5\}, \{1, 1.0\}, \{2, 1.0\}\}.$$

From this example, it can be seen that two overlapping occurrences of the pitch-interval sequence are present at $\mathcal{F}_1$. In this circumstance, it is very desirable for the second occurrence of the pitch interval sequence to be bound to its rhythm representation at the other SONNET module. However, this would not be possible using the current formulation for handling phrase repetitions at $\mathcal{F}_2$. This is because the current scheme does not provide the framework for handling repetitions of phrases that overlap. Therefore, the second occurrence of the phrase will never be recognized. Consequently, the pitch-interval sequence will not be bound to its corresponding rhythm unless both patterns occur again in a different, more suitable context.

Both of these issues are serious drawbacks and need to be resolved. This can be achieved by utilizing a special form of competition between cell assemblies that classify the same pattern ([114, Section 6.11]). This competition must take place, such that cell assemblies within the same sibling group at $\mathcal{F}_2$ *do not respond identically* to input items presented at $\mathcal{F}_1$. This competition must not take place between the cell assemblies themselves, either via lateral inhibition between classification cells or latch cells, for the reasons just described. Instead, a more selective form of competition must take place. This competition should take place between the *links* entering different cell assemblies (within the same sibling group at $\mathcal{F}_2$) rather than between the *cells* themselves. This approach will enable the repetition of a phrase to be processed (including an overlapping repetition) as soon as it occurs, rather than relying on previous occurrences of the phrase being completely processed first. In other words, no two siblings, within the same sibling group at $\mathcal{F}_2$, may begin their trajectory at the same time, in response to the same note onset.

Remember, links emanating from the j^{th} sibling group at $\mathcal{F}_1$ to a particular cell assembly at $\mathcal{F}_2$, compete (via intra-cell competition) to enable item repetitions to be processed. In contrast, links that originate from a particular input assembly at $\mathcal{F}_1$, and terminate at the i^{th} sibling group at $\mathcal{F}_2$ should compete (via inter-cell competition); this time to enable phrase repetitions to be

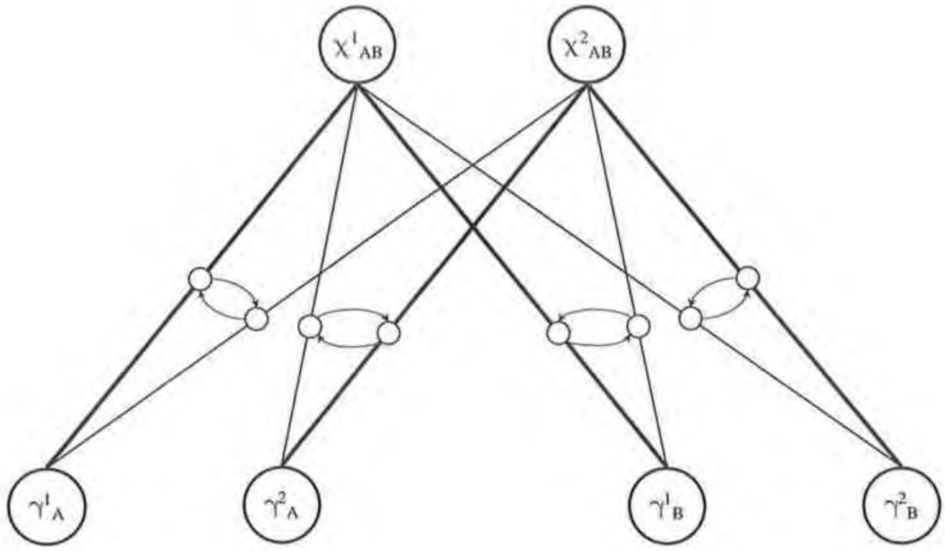


Figure 8.2: Inter-cell competition. χ^1_{AB} and χ^2_{AB} must not respond identically to event occurrences across $\mathcal{F}_1$. In this example, χ^1_{AB} responds to the first occurrence of {A, B}, while χ^2_{AB} responds to the second occurrence. Inter-link competition should enable χ^2_{AB} to respond to the second occurrence of the phrase {A, B} as soon as it begins.

processed. For example, χ^1_{AB} (or χ^2_{AB}) should respond to the first occurrence of its phrase in the pattern {A, B, A, B}, while χ^2_{AB} (or χ^1_{AB}) should begin responding to the second repetition as soon as it occurs, even in the the case when identical phrases overlap (see Figure 8.2). The manner by which this competition is achieved will be described next.

8.2.3 Using feedback to implement presynaptic inter-cell competition

Section 6.2 showed how the use of multiple interacting links enabled SONNET to process repeated items. The interactions were implemented by a form of feed-forward competition, whereby only the activity of the links themselves were utilized (see Figure 8.3(a) and 8.4(a)). In this circumstance, the activity level of each link provided enough information to enable sufficient contrast enhancement to occur. This is because, in this circumstance, the competing links emanated from input cells at $\mathcal{F}_1$ with different activity levels. However, when links from the same input assembly at $\mathcal{F}_1$, which terminate at different $\mathcal{F}_2$ cell assemblies (within the same sibling group), compete to enable phrase repetitions to be processed, link activity does not provide enough information. This is because, in this circumstance, the competing links emanate from the same input cell at $\mathcal{F}_1$. Therefore, to allow sufficient contrast enhancement, *only feedback from the link's cell assembly will be used to implement competition* (see Figure 8.3(b) and 8.4(b)). To facilitate this new competition between links, additional network structure is required (see Figure 8.5):

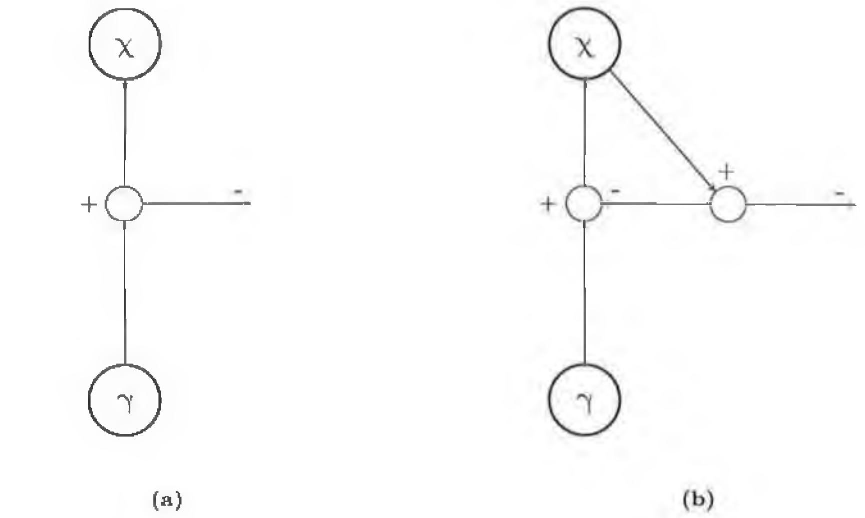


Figure 8.3: (a) Feed-forward competition enables item repetitions to be processed. (b) Feedback competition enables phrase repetitions to be processed.

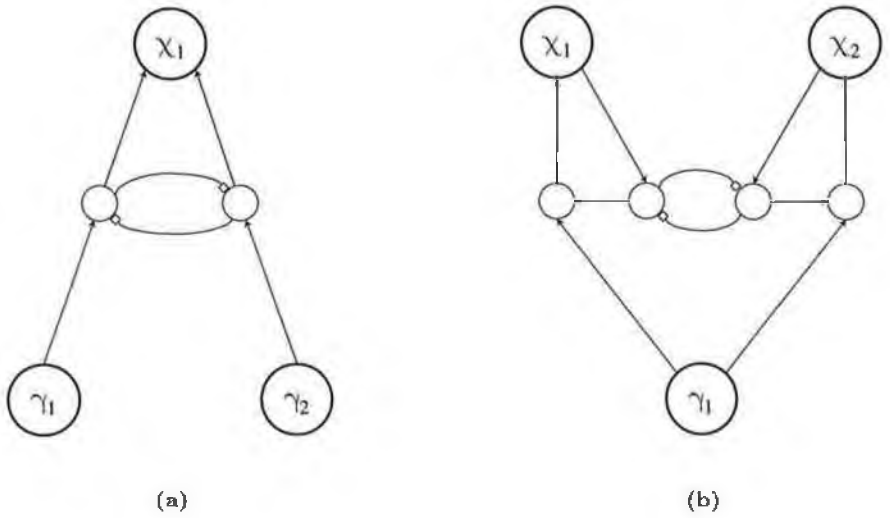


Figure 8.4: (a) Intra-cell competition utilizes feed-forward competition. (b) In contrast, inter-cell competition utilizes feedback competition.

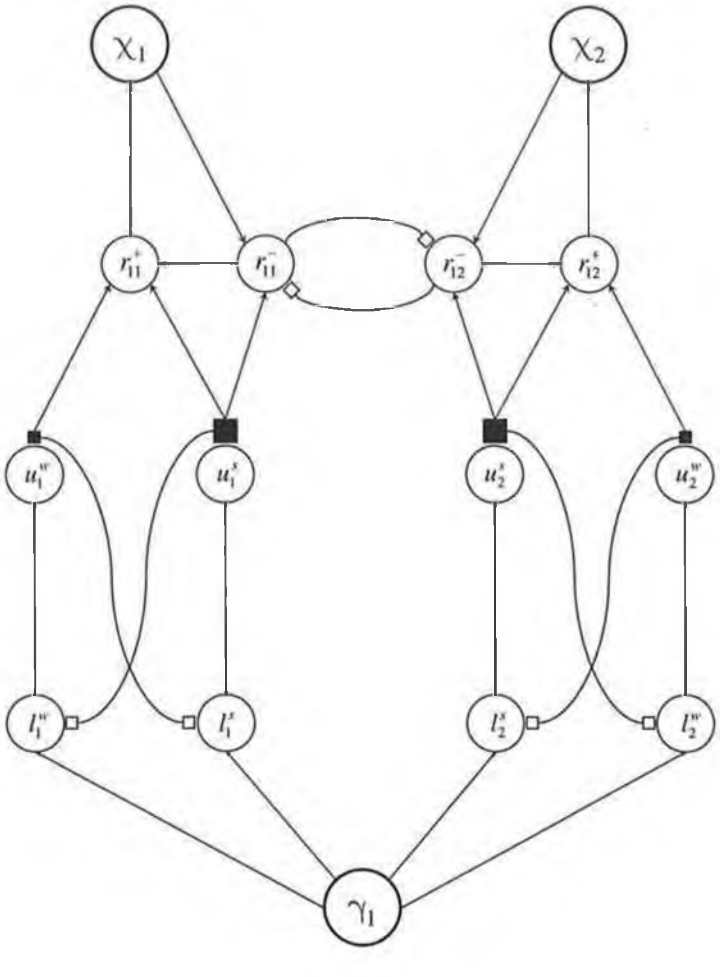


Figure 8.5: Inter-cell architecture showing the links required to process item repetitions and phrase repetitions. Note that, since only a single input assembly and a single sibling group at $\mathcal{F}_2$ is illustrated, the m and n indexes are dropped.

Feed-forward relay cells. Previously, the activity from the links of each link cluster[2] fed directly into their corresponding excitatory cell. However, since it is unclear how the interactions between inter-cell competition and intra-cell competition would behave, all excitatory signals from a link cluster will be received by a single feed-forward relay cell (FFRC), labelled r^{+}_{mjni}, instead. This has the effect of separating the processing of item repetitions from the processing phrase repetitions. Therefore, item repetition handling will behave in exactly the same manner as described before. Furthermore, an entire link cluster can be shut-off if it receives an inhibitory signal from its associated feedback relay cell (FBRC), thereby shutting-off any excitatory input to its corresponding cell assembly. In effect, inter-cell competition can be viewed as occurring between link clusters rather than individual links.

Feedback relay cells. A feedback relay cell (FBRC), labelled r^{-}_{mjni}, receives an excitatory input signal from its corresponding cell assembly's feedback cell, d_{ni}, and a signal from the link cluster's strongest link. FBRCs may only compete with one another if, (1) they originate from the same input assembly at $\mathcal{F}_1$, (2) they terminate at different cell assemblies within the same $\mathcal{F}_2$ sibling group and (3) the active link in the cluster is the strongest link. Such FBRCs are mutually inhibitory, therefore only a single link cluster from γ_{mj} to the i^{th} sibling group at $\mathcal{F}_2$ may activate. Initially, in response to an event occurrence at γ_{mj}, each link cluster is as likely to win this competition as any other. Therefore, each FBRC activation is gated by an inhibitory weight from every other FBRC; this facilitates the emergence of a clear winner among the competing links. These weights are initialized over a narrow range and enable one of the FBRCs to gain an initial competitive advantage. These inhibitory weights act as a repeat selector, similar to that described by Bradski et al. [11]. The third condition specified above enables overlapping phrase repetitions to be recognized.

8.2.4 Rules governing inter-cell competition

In addition to the structure describes so far, the following rules govern the operation of the inter-cell competition used to enable phrase repetitions to be processed:

1. Link clusters may only compete if their strongest link is active. A FBRC can determine if the strongest link in its associated link cluster is active, by directly examining the output from this link (see Figure 8.5).

2. Any FBRC that loses the competition will shut-off its corresponding FFRC and, consequently, reset the entire link cluster. The entire link cluster will remain shut-off until its corresponding input cell at $\mathcal{F}_1$ has been reset.

3. Inter-cell competition may only take place within a sibling group when that sibling group has committed to a pattern (that is, $\chi^{com}_{ni} = true$). To achieve this, initially only a single

[2] Remember, the term link cluster was introduced in Chapter 6 to describe the set of N_l links between each $\gamma-\chi$ pair.

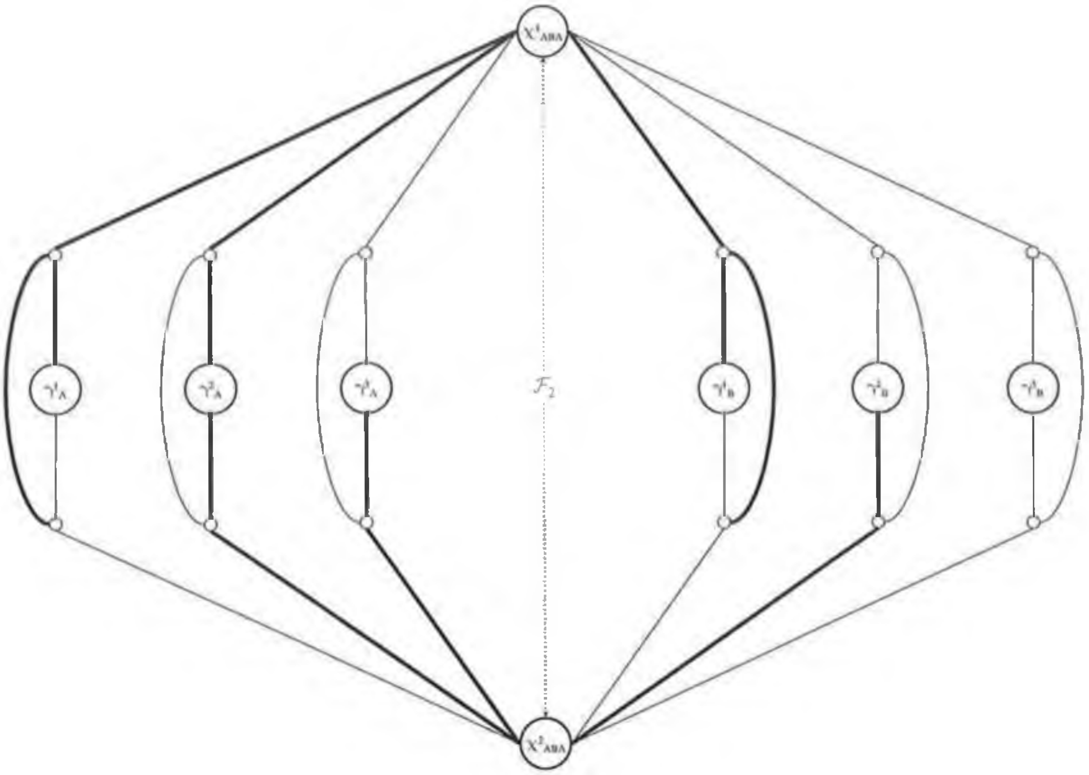


Figure 8.6: Recognizing overlapping phrase repetition: {A, B, A, B, A}.

cell assembly within each group may respond to events at $\mathcal{F}_1$. Then, when this cell assembly commits to a pattern, the entire group is considered to have become committed too. In this circumstance, any cell assembly within the sibling group at $\mathcal{F}_2$ may respond to patterns across $\mathcal{F}_1$.

8.2.5 Example

The behaviour of the presynaptic inter-cell interactions will now be explained with reference to Figure 8.6, which illustrates the competition between link clusters from $\mathcal{F}_1$, representing occurrences of the items A and B, and cell assemblies at $\mathcal{F}_2$, representing occurrences of the learned sequence {A, B, A}. In this example, $N^s_1 = 3$, $N^s_2 = 2$ and the input pattern {A, B, A, B, A} is presented to $\mathcal{F}_1$, which represents two overlapping occurrences of the pattern {A, B, A}.

Item A When the first event occurs, any sibling within the group representing the item A may activate. In this example, γ^1_A is activated. Firstly, in this circumstance, both the link clusters $\gamma^1_A - \chi^1_{ABA}$ and $\gamma^1_A - \chi^2_{ABA}$ will compete among themselves in the normal way, in order to process item repetitions at $\mathcal{F}_1$. The strongest link in each link cluster will activate and propagate their excitatory signals to their respective FFRCs. These signals are then forwarded to their respective cell assemblies χ^1_{ABA} and χ^2_{ABA}. Secondly, at this point, due to the activation of these cell assemblies, the FBRCs will receive excitatory signals from their corresponding feedback cells (d^1_{ABA} and d^2_{ABA} respectively). Since the strongest link is active in both link clusters, inter-cell competition is initiated between χ^1_{ABA} and χ^2_{ABA}. Initially, since the excitatory input to the FFRCs will emanate from the strongest link in each link cluster, and since the signals on these links will be identical (same input and excitatory weight values), there will be little or no difference between the feedback signals received by the FBRCs. Consequently, in this circumstance, the inhibitory weights between the FBRCs will be utilized to resolve the winning link. Suppose, for the current example, that the link cluster $\gamma^1_A - \chi^1_{ABA}$ wins and suppresses the link cluster $\gamma^1_A - \chi^2_{ABA}$. This will cause the entire link cluster $\gamma^1_A - \chi^2_{ABA}$ to be reset. Since χ^2_{ABA} will no longer receive an excitatory signal, its activation will decay back to zero.

Item B When the second event occurs, any sibling within the group representing the item B may activate. In this example, γ^1_B is activated. Firstly, in this circumstance, both the link clusters $\gamma^1_B - \chi^1_{ABA}$ and $\gamma^1_B - \chi^2_{ABA}$ will compete among themselves in the normal way, in order to process item repetitions at $\mathcal{F}_1$. The strongest link in each link cluster will activate and propagate their excitatory signals to their respective FFRCs. These signals are then forwarded to their respective cell assemblies χ^1_{ABA} and χ^2_{ABA}. Secondly, at this point, due to the current activation of these cell assemblies, the FBRCs will receive excitatory signals from their corresponding feedback cells (d^1_{ABA} and d^2_{ABA} respectively). This initiates the inter-cell competition between χ^1_{ABA} and χ^2_{ABA}. In this circumstance however, χ^1_{ABA} will have a higher activation than χ^2_{ABA}. Due to χ^1_{ABA}'s competitive advantage at this point, the link cluster $\gamma^1_B - \chi^2_{ABA}$ will be reset.

Item A When the third item occurs (which is the second occurrence of the item A), the last item of χ^1_{ABA}'s pattern and the beginning of the second phrase repetition, either γ^2_A or γ^3_A may activate. In this example, γ^2_A is activated. In this circumstance, the weakest link in the link cluster $\gamma^2_A - \chi^1_{ABA}$ will be active. Furthermore, since the link cluster $\gamma^1_A - \chi^2_{ABA}$ has already been shut-off, the strongest link will become active in the link cluster $\gamma^2_A - \chi^2_{ABA}$. Consequently, these link clusters will not compete. In this situation, χ^1_{ABA} will completely recognize the first occurrence of the phrase {A, B, A} across $\mathcal{F}_1$. Furthermore, χ^2_{ABA} will be responding to the beginning of the second occurrence of the phrase {A, B, A}.

Item B When the fourth item occurs, which is the second occurrence of the item B and the second item in the second phrase repetition, either γ^2_B or γ^3_B may activate. In this example, γ^2_B is activated. Since χ^1_{ABA} has already recognized its pattern and, since the link cluster $\gamma^1_B - \chi^2_{ABA}$ has already been shut-off, the strongest link in $\gamma^2_B - \chi^2_{ABA}$ will activate.

Item A When the fifth item occurs (which is the third occurrence of the item A and the final item of the second phrase repetition), γ^3_A will activate. Again, since χ^1_{ABA} has already recognized its pattern, the weakest link in $\gamma^3_A - \chi^2_{ABA}$ will activate, thus leading to the complete recognition of the second phrase by χ^2_{ABA}.

This section described the application of the link interactions in enabling the repetition of phrases to be recognized and correctly processed. The manner by which this mechanism can be modelled and implemented is described in the following section.

8.2.6 Modelling the presynaptic inter-cell interactions

To model the behaviour described thus far, only *active* cell assemblies may modify their cell activations and adjustable weights before a sibling group at $\mathcal{F}_2$ has become committed. A cell assembly is defined as being active if $\chi_{ni}^{active} = true$. Initially, only one cell assembly within each sibling group may be active. This remains true until that cell assembly learns, and commits to, a pattern, thereby committing the entire sibling group to that pattern. This ensures that no inter-cell competition take place unless that sibling group represents a pattern (see Section 8.2.4). Note that if $N_2^s = 1$, no phrase repetition handling needs to occur.

When a cell assembly, χ_{ni}, becomes committed, inter-cell competition may occur by simply activating additional cell assemblies within the sibling group at the appropriate times and in the appropriate manner. It is the means by which this activation occurs that enables the link interactions that have been described so far to be modelled. Either of two approaches can be taken when activating an additional cell assembly:

1. An additional cell assembly within a sibling group may become active only when one of the other active cell assemblies is responding to the beginning of its pattern. It can be determined that a cell assembly is responding to the beginning of its pattern if it is on-target. That is, if $I_{ni}^{\times} \approx W_{ni}^{\times}$. For SSG architectures encoding interval information, it must be ensured that, in addition to being on-target, the cell assembly must be responding to at least two onsets. This is because at least two onsets are required to encode a single interval.

2. An additional cell assembly within a sibling group may become active only when (a) one of the other active cell assemblies is responding to its entire pattern, and (b), when an event occurs which is part of the the i^{th} sibling group's learned pattern. It can be determined that a cell assembly is responding to its entire pattern if $I_{ni}^{\times} \geq K_M$.

Both approaches will achieve the behaviour described thus far. However, in this thesis, the second approach is used. This is because, using the first approach, it is possible for numerous cell assemblies to become active (particularly for SSG architectures) in response to input patterns at $\mathcal{F}_1$. This is due to the fact that only the beginning of a pattern needs to be recognized for additional cell assemblies to be activated. This is undesirable because it is computationally more expensive. Note, in the case of a hierarchy, a cell assembly may only become active if its corresponding input cell in the layer above is inactive.

When a cell assembly becomes active it must, at the very least, respond to events that occurred after the oldest onset represented by the cell assembly that triggered the activation of the additional cell assembly. For example, consider the state of $\mathcal{F}_1$ in response to the input pattern {A, A, A, B, A, B} (see Figure 8.7). In this example, the pattern {A, B} has been learned by the i^{th} sibling

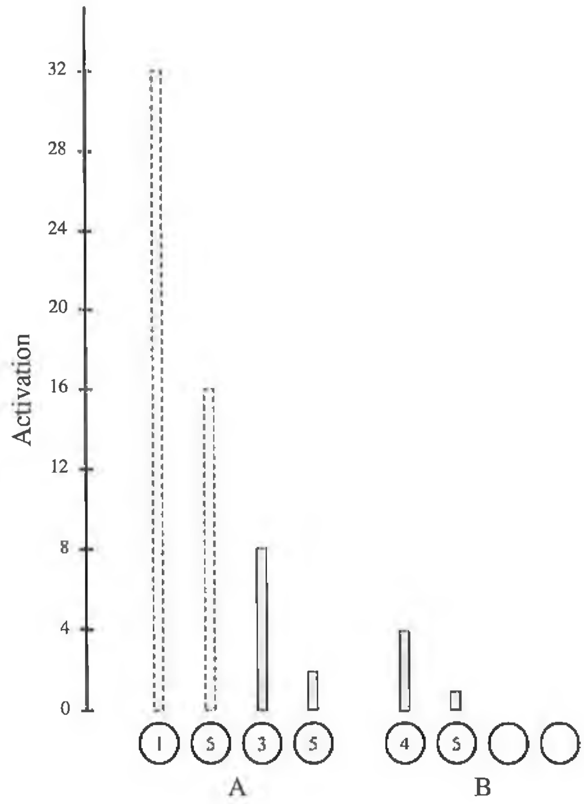


Figure 8.7: The activating cell assembly should respond to event onsets 3 and 4, whereas the newly activated cell assembly should respond to event onsets 5 and 6.

group at $\mathcal{F}_2$, which consists of two siblings per group ($N_2^s = 2$). Furthermore, from Figure 8.7, it can be seen that two occurrences of the phrase {A, B} are present at $\mathcal{F}_1$, events $3 - 4$ and events $5 - 6$ respectively. Also note that the first two events, {A, A}, do not form part of the i^{th} sibling group's pattern. Consequently, the links from the event onsets 1 and 2 to χ_{AB}^1 will have been reset due to the relay-cell reset mechanism. In this example, by the time the fifth event has occurred, χ_{AB}^1 will be responding to its entire pattern and the beginning of the second phrase repetition will be present. Consequently, χ_{AB}^2 will be activated in preparation for the occurrence of the second phrase repetition. Since χ_{AB}^2 has not been active until now, initially it would respond to events $1 - 2$ because no relay cell resets would have occurred. This is not the desired behaviour. We require χ_{AB}^2 to respond to the event onsets that occur subsequent to those onsets that χ_{AB}^1 responds to. That is, onsets $5 - 6$. To achieve this, the number of relay-cell resets from each of the $\mathcal{F}_1$ sibling groups (which are elements of T_{AB}^1) to the activating cell assembly must be propagated to the newly-activated cell assembly. At this point, the newly-activated cell assembly would now be responding to the same pattern as the activating cell assembly. Therefore, an additional reset is also triggered from each sibling group at $\mathcal{F}_1$. This enables the newly-activated cell assembly to respond to later events as required. In this case, events 5 and 6. The number of reset links, n_{jmi}, from the j^{th} sibling group at $\mathcal{F}_1$ to a newly-activated cell assembly, χ_{mi}, is given as follows:

$$n_{jmi} = n_{jni} + 1 \tag{8.1}$$

where n_{jni} represents the number of resets in the activating cell assembly.

Once a cell assembly has caused an additional cell assembly within its sibling group to activate, it may not cause another cell assembly to activate until after it has become reset, and subsequently becomes active again. When a cell assembly is reset to zero and becomes inactive; χ_{ni}^{active} = false. Note also, when a cell assembly becomes inactive, it must be ensured that there is at least one other active cell assembly within its sibling group available to respond to input patterns at $\mathcal{F}_1$. Finally, since all cell assemblies within a sibling group encode the same pattern, there are only $|T_{ni}|$ weights. The weights from $\mathcal{F}_1$ to $\mathcal{F}_2$ are indexed in exactly the same way as before. For example, the link from γ_{mj} to χ_{ni} is referenced by z_{kjni}^{+}.

In summary, an active, committed cell assembly which matches its pattern may cause another cell assembly within its sibling group to become active. Once a cell assembly has activated another cell assembly, it may not activate another unless it has been reset (either by the chunking-out process or by one of its active links becoming reset). Relay cell resets are propagated from the activating cell assembly to the newly-activated cell assembly. In addition, one further link is also reset from each sibling group at $\mathcal{F}_1$. This enables the newly-active cell assembly to respond to the appropriate event onsets.

8.3 A new multiplicative $I_{ni}^{\times}$ formulation

$I_{ni}^{\times}$ quantifies the match between the normalized weight vector and the normalized input vector specified by χ_{ni}'s receptive field T_{ni}. In other words, $I_{ni}^{\times}$ measures how well χ_{ni} represents the current input pattern at $\mathcal{F}_1$. Since excitatory weights converge to decreasing activity patterns (see Section 5.3.1), weights that code for items near the end of a sequence are relatively small compared with those that code for items near the beginning of a sequence. $I_{ni}^{\times}$ allows the presence or absence of input activities that are gated by small weights to have a significant effect on the total input to χ_{ni}. This ability is crucial to solving the temporal chunking problem because, in addition to affecting total input to χ_{ni}, the size of a cell assembly D_{ni} is directly related to $I_{ni}^{\times}$ and, as Sections 5.3.5 and 5.3.6 have shown, D_{ni} dilutes both the excitatory and inhibitory inputs to χ_{ni}.

To achieve this behaviour, a confidence formulation must posses the following properties:

Property 1. Larger cell assemblies should have higher maximum confidence values than smaller cell assemblies. This enables larger cell assemblies to mask-out smaller cell assemblies.

Property 2. When a cell assembly's pattern is not completely present at $\mathcal{F}_1$, its confidence value must accurately reflect the level of mismatch between its normalized weight vector and its normalized input vector.

Property 3. $I_{ni}^{\times}$ should increase with a decrement in $|T_{ni}|$ to avoid a spurious termination of learning at χ_{ni}.

Input	χ_{AB}	χ_{ABC}	χ_{ABCD}
A	0.92	0.45	0.23
AB	4	1.93	0.96
ABC	4	8	3.95
ABCD	4	8	16

Table 8.1: Various confidence values obtained with $K_1 = 0.5$, $K_2 = 1.5$ and $K_8 = 0.05$. For example, the confidence value of $\chi_{ABC} = 1.93$ when the pattern {A, B} is active at $\mathcal{F}_1$.

8.3.1 Previous $I^{\times}_{ni}$ formulations

Nigrin's original specification, with improvements by Page (expectation) and Roberts (to avoid spurious termination of learning), is repeated here for convenience (with indices for multiple sibling groups):

$$I^{\times}_{ni} = \max\left(\prod_{k,j \in T_{ni}} I^{\times}_{kjni}, K_8\right) \tag{8.2}$$

$$I^{\times}_{kjni} = K_1 + K_2 \min\left[\left(\frac{Z_{kjni}}{S_{kjni}}\right)^{f_{ni}}, \frac{S_{kjni}}{Z_{kjni}}\right] \tag{8.3}$$

$$f_{ni} = \frac{1}{1 + e^{100(K_Z - z^{tot}_{ni})}} \tag{8.4}$$

This formulation satisfies *Property 1*. For example, suppose the patterns {A, B}, {A, B, C}, {A, B, C, D} have each been learned by χ_{AB}, χ_{ABC}, χ_{ABCD} respectively. Furthermore, suppose the pattern {A, B, C, D} is present at $\mathcal{F}_1$. Table 8.1 shows the various confidence values at each cell assembly. From Table 8.1, it can be seen that larger cell assemblies, which respond to their complete pattern, have higher confidence values than smaller cell assemblies, which also respond to their complete pattern. For example, $I^{\times}_{ABCD} = 16$ ($I^{\times}_{ABCD}(max) = (K_1 + K_2)^{|T_{ABCD}|}$), whereas $I^{\times}_{ABC} = 8$.

Property 3 is also satisfied. Suppose the input sequence {A, B, C} is repeatedly presented to a SONNET module. In this circumstance, a number of cell assemblies will compete to encode the pattern {A, B, C}. This will lead to significant links being formed from the input assemblies at $\mathcal{F}_1$ representing A, B and C to these cell assemblies. Eventually one of these cell assemblies will completely learn the pattern and become committed to it. Then, there will be a number of cell assemblies with partial learning of the pattern {A, B, C}. Now, suppose the input sequence {A, B} is repeatedly presented. In this circumstance, it is likely that at least one of these partially encoded cell assemblies will compete to learn the smaller pattern {A, B}. For convenience we label this cell assembly $\chi_{partial}$. Since $\chi_{partial}$ has previously competed to learn the pattern {A, B, C}, its confidence value will be equal to 1.93 when {A, B} is present at $\mathcal{F}_1$ (see Table 8.1). Suppose that $\chi_{partial}$ was to learn the pattern {A, B}, then the link from the input assembly γ_C to $\chi_{partial}$ would be removed from $T_{partial}$, thus decrementing $|T_{partial}|$. In this circumstance, $I^{\times}_{partial}$ will increase to 4. Thus, $I^{\times}_{partial}$ increases with a decrement in $|T_{partial}|$ satisfying *Property 3.*

Due to the K_8 parameter in Nigrin's original confidence formulation, *Property 2* is not always achieved. K_8 is required to prevent $I^{\times}_{ni}$ from becoming arbitrarily small, which would cause very slow learning. The difficulty with K_8 is particularly evident for cell assemblies that respond to long patterns. For example (see Nigrin [114, Section 3.16.3]), suppose χ_{ni} receives input from 25 input assemblies. Furthermore, suppose that the absence of 10 of these input assemblies causes $I^{\times}_{ni}$ to reduce to K_8. Whenever $I^{\times}_{ni}$ is reduced to this floor value it is not possible to tell how much of a cell assembly's pattern is present or absent. This leads to ambiguities when comparing confidence values. For example, vigilance is hampered during inhibitory weight learning. Although with judicious parameter setting these problems may be overcome, they tend to be application specific and will not apply generally. Therefore it is desirable to remove the K_8 parameter from the confidence formulation altogether.

Other multiplicative confidence formulations have also been previously introduced. However, each formulation fails to satisfy at least one of the properties outlined above. For example, the formulation presented by Nigrin [114, Section 3.16.3] satisfies *Property 1* and *Property 2* but fails to satisfy *Property 3.* Additionally, the formulation presented by Roberts [133, Section 5.8.1] satisfies *Property 1* and *Property 3* but fails to satisfy *Property 2.*

8.3.2 Multiplicative verses additive $I^{\times}_{ni}$ formulations

Nigrin's confidence formulation increases exponentially with $|T_{ni}|$. Consequently, this formulation favours the encoding of the largest possible pattern. This is advantageous because larger patterns give more predictive power and lead to better sequence compression. Additionally, larger patterns have a greater discriminatory power when retrieval is the primary goal. However, Roberts observed that, for modelling rhythm perception, small patterns were discriminated against, which could lead to inefficient sequence segmentations. To address this problem, Roberts [133, Section 6.3] introduced an additive confidence formulation (which satisfied all of the properties listed above):

$$I^{\times}_{ni} = 1 + \frac{K_2}{|T_{ni}|}\left(\sum_{k,j\in T_{ni}} I^{\times}_{kjni}\right)^2 \qquad (8.5)$$

$$I^{\times}_{kjni} = \min\left(\frac{Z_{kjni}}{S_{kjni}}, \frac{S_{kjni}}{Z_{kjni}}\right) \qquad (8.6)$$

Although Roberts shows how this formula can solve the temporal chunking problem and allow context sensitive recognition (for example χ_{AB} and χ_{CD} can combine to mask-out χ_{ABC} when {A, B, C, D} is present at $\mathcal{F}_1$), Roberts analysis is based upon the following assumptions:

1. All classification cell activations are identical: $c_{ni} = c < 1$.

2. All cell assemblies have been active for the same length of time: $\tau_{ni} = \tau$.

These assumptions are only true when all inputs are presented simultaneously (that is, a spatial pattern), which is not the case when sequences are presented item by item in real-time. Through extensive preliminary SONNET simulations, it was discovered that this formulation often suffers from the following problems:

- Small patterns will generally overpower larger patterns if they overlap with each other. This is because c_{ni} will become sufficiently large, along with $I^{\times}_{ni}$, to overpower any of its competitors regardless of their size. Therefore, although this formulation does not discriminate against small patterns, it does discriminate against large patterns and can therefore lead to inefficient sequence segmentations.

- Once smaller sub-patterns are learned, it is extremely difficult or even impossible for larger patterns that contain these sub-patterns to be learned (see Roberts [133, Section 6.8.2] test results). Thus in general, this formulation fails to solve the temporal chunking problem.

Although this formulation can work well for modelling rhythm perception, in this thesis we favour the use of a multiplicative confidence formulation. Particularly one which satisfies all of the properties listed above. Such a formulation will be introduced next.

8.3.3 The new confidence formulation

This section introduces a new multiplicative confidence formula, which satisfies *Property 1*, *Property 2* and *Property 3*:

$$I^{\times}_{ni} = \left[K_{\times} + \left(\frac{I^{*}_{ni}}{|T_{ni}|}\right)^{2}\right]^{I^{*}_{ni}} \tag{8.7}$$

$$I^{*}_{ni} = \sum_{k,j\in T_{ni}} I^{\times}_{kjni} \tag{8.8}$$

$$I^{\times}_{kjni} = \min\left(\frac{Z_{kjni}}{S_{kjni}}, \frac{S_{kjni}}{Z_{kjni}}\right) \tag{8.9}$$

where $K_{\times}$ represents the minimum value that $I^{\times}_{ni}$ may reach, $K_{\times} \geq 1$. Furthermore, this parameter also helps define the maximum $I^{\times}_{ni}$ a cell assembly may obtain. This is true because when a cell assembly matches its input at $\mathcal{F}_1$, the term $\frac{I^{*}_{ni}}{|T_{ni}|}$ reaches unity. This is due to the fact that the division of I^{*}_{ni} by $|T_{ni}|$ is effectively a normalization process. Thus, the maximum confidence value a cell assembly may reach is $(K_{\times}+1)^{|T_{ni}|}$.

With a typical value of $K_{\times} = 1$, Table 8.2 illustrates possible confidence values. By examining the table it can be seen that *Property 1* is satisfied because larger cell assemblies always have larger confidence values than smaller cell assemblies when they respond to their complete patterns at $\mathcal{F}_1$. Furthermore, *Property 2* is also satisfied. This is because there is no parameter (K_8) required for preventing the confidence values from becoming arbitrary small. As a result, if no input is present, $I^{\times}_{ni}$ reduces to its minimum value, which is unity. Furthermore, if any of a cell assembly's inputs are present, $I^{\times}_{ni}$ will always be greater than unity. Finally, since $I^{\times}_{ni}$ increases with a decrement in $|T_{ni}|$, *Property 3* is satisfied too.

Input	χ_{AB}	χ_{ABC}	χ_{ABCD}
No Input	1	1	1
A	1.18	1.07	1.04
AB	4	1.99	1.51
ABC	4	8	3.74
ABCD	4	8	16

Table 8.2: Various confidence values obtained with $K_\times = 1$. For example, the confidence value of $\chi_{ABCD} = 3.74$ when the pattern {A, B, C} is active at $\mathcal{F}_1$.

Since $I^\times_{ni}$ will never fall below unity, the vigilance formulation is modified slightly:

$$\rho_{ni} = (I^\times_{ni})^\rho \tag{8.10}$$

Additionally, the quantity $W^\times_{ni}$, used in Page's expectancy work (see Chapter 7), should be analogous to this new confidence formulation. Therefore, if this expectancy work is to be used with the new formulation described above, the following $W^\times_{ni}$ formulation should be used:

$$W^\times_{ni} = \left[K_\times + \left(\frac{W^*_{ni}}{|T_{ni}|}\right)^2\right]^{W^*_{ni}} \tag{8.11}$$

$$W^*_{ni} = \sum_{k,j \in T_{ni}} W^\times_{kjni} \tag{8.12}$$

$$W^\times_{kjni} = \begin{cases} 1 & \text{if } S_{kjni} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{8.13}$$

(8.14)

8.4 Repeating previous SONNET experiments

In order to verify the implementation, operation and behaviour of the SONNET network with the new modifications described in this chapter, the simulations described by Nigrin [114, Chapter 4], Page [118, Chapter 7] and Roberts [133, Chapter 6] were reproduced. This section briefly describes the results we obtained.

8.4.1 Nigrin's embedded phrase learning experiments

Nigrin [114, Section 4.6] reported simulations that verified the behaviour of his SONNET specification. In these simulations, isochronous sequences consisting of either letters or numbers were presented to SONNET using an event-driven SONNET configuration (see Appendix A). The task was to discover any sequential regularities inherent in these input sequences.

These simulations were reproduced, using the same parameter values as used by Nigrin [114, Appendix C], and the results were practically identical. Although minor variations existed due to randomness inherent in the network (see below), the desired sequential regularities were discovered in every case. For example, ten epochs of the following sequence were presented to SONNET in order to verify if the significant patterns {0, 1, 2} and {24, 25, 26} could be learned (see Simulation 3 [114, Section 4.6])

0	1	2	3	4	5	24	25	26	6	7	8	9
0	1	2	10	11	12	13	24	25	26	14	15	16
0	1	2	17	18	19	24	25	26	20	21	22	23

In our simulations, the pattern {0, 1, 2} was learned on the 5^{th} epoch, while the pattern {24, 25, 26} was learned on the 6^{th} epoch.

One point emerged during these experiments, with respect to the use of randomness in the network. Randomness is present in two forms. Firstly, in the initialization of the excitatory weights, and secondly, in the use of a random element in the $\mathcal{F}_1$ overload scheme. It was observed that the behaviour of the network was somewhat dependent on how the network was initialized and the manner in which the random element in the $\mathcal{F}_1$ overload scheme operated. Consequently, minor differences between simulation runs were observed. For example, differences arose in the number of epochs required to learn each pattern.

8.4.2 Page's melody learning task and expectancy generation experiments

Section 7.2 described research by Page that showed how a SONNET hierarchy could learn simple nursery rhyme melodies and how feedback could be utilized to enable melody recall. This section reproduces Page's hierarchical learning and expectation experiments [118, Section 7.3]. As far as we are aware, this is the first time that Page's expectancy work has been reproduced.

Learning the nursery rhymes

Page trained SONNET using an input set consisting of twelve simplified nursery rhyme melodies. The melodies were represented as sequences of diatonic scale notes each lasting the duration of an attentional pulse (in Page's experiments, $T_{attn} = 0.2$). Each note occurred in synchrony with this pulse. Consequently, Page used an event-driven SONNET configuration (see Section 6.3.2). Additionally, some notes at the end of a phrase, which last two beats, enjoy an extended learning period. The nursery rhyme *Boys and Girls Come Out to Play*, for example, would be represented as follows:

$$\{\{5, 4, 5, 1\}\ \{2, 4, 5, 1\}\ \{5, 4, 5, 1\}\ \{2, 4, 5, 1\}\}$$

To learn the nursery rhymes, SONNET was configured as a hierarchy. The first module learned familiar phrases at the note level while the second module learned familiar phrase sequences (phrases of phrases). The learning of phrase sequences at $\mathcal{F}_3$ may only occur when the phrases themselves are sufficiently familiar to have been learned by cell assemblies at $\mathcal{F}_2$. Consequently, the learning of phrase sequences was disabled until the constituent phrases of each melody were learned. When this occurs, learning at the higher level was enabled.

$\mathcal{F}_1$ consisted of 7 sibling groups, each containing 4 siblings; $\mathcal{F}_2$ consisted of 45 sibling groups, each containing 4 siblings; while $\mathcal{F}_3$ consisted of 30 sibling groups with a single sibling in each group. Each melody was presented incrementally to SONNET. With the *incremental* presentation regime, a single melody is presented repeatedly, until it is completely learnt by SONNET, before the next melody is presented. Ten epochs of each melody was presented in each case, which was more than adequate to learn the phrases comprising each melody. Then, each melody was presented an additional ten times in order for the phrase sequence of each melody to be learned. The network was trained using the same parameters used by Page. In particular, no random element was used in the $\mathcal{F}_1$ overload mechanism. Instead, when g_s reached the overload threshold, K_{12}, the four oldest events were deleted from $\mathcal{F}_1$

Note that, due to the fortuitous structure of each melody and the $\mathcal{F}_1$ overload mechanism employed, this scheme should be considered as processing pre-segmented input. This is because each phrase consisted of four beats, and the overload mechanism deleted the four oldest events when the K_{12} threshold was reached. Indeed, Page [118, pg. 237] points out that *'in general, we may not be able to rely on judicious settings of fixed $\mathcal{F}_1$ reset parameters to "discourage" the learning of "across-phrase" groupings from continuous presentations, and that the alternative "clues" to the likely location of phrase boundaries might thus have to be invoked'.* In fact, the work presented in Section 7.3 represents the first step towards using such alternative "clues". Nonetheless, irrespective of the means by which SONNET is populated with melodic phrases, expectancies are generated in the same way.

Melody recall using a SONNET hierarchy

Once each of the twelve melodies was learned, melody recall was tested. To achieve this, each melody was effectively *named*. Then the task was to produce expectancies consistent with the pitch sequence of the named melody. Naming a melody was achieved by activating the feedback cell, and pegging it to unity, at the cell assembly representing the desired melody at $\mathcal{F}_3$. The expectations generated at $\mathcal{F}_1$ were then recorded before each note of the melody was presented. This processes continued until the entire melody was presented. In every case the most expected item was always the next note in the melody. In effect, the network was capable of perfectly recalling every melody.

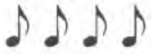
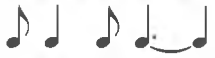
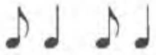
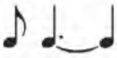
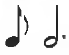
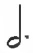
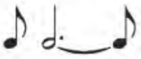
For example, the expectancies generated when the cell assembly at $\mathcal{F}_3$, representing the melody *Boys and Girls Come Out to Play*, had its feedback cell pegged at 1, are presented in Table 8.3. The expectancies obtained are consistent with the results obtained by Page [118, Table 7.8].

Melody	{{5, 4, 5, 1} {2, 4, 5, 1} {5, 4, 5, 1} {2, 4, 5, 1}}				
Items at $\mathcal{F}_1$	Expectations at $\mathcal{F}_1$				
—	5(0.261)	4(0.130)	5(0.065)	1(0.031)	—
5	4(0.157)	5(0.079)	1(0.050)	2(0.050)	3(0.027)
5,4	5(0.097)	3(0.070)	1(0.049)	6(0.048)	2(0.033)
5,4,5	1(0.072)	—	—	—	—
—	2(0.261)	4(0.131)	5(0.065)	1(0.033)	6(0.001)
2	4(0.157)	5(0.079)	3(0.051)	1(0.039)	2(0.026)
2,4	5(0.099)	1(0.049)	3(0.033)	—	—
2,4,5	1(0.072)	3(0.016)	—	—	—
—	5(0.261)	4(0.131)	5(0.065)	1(0.033)	3(0.001)
5	4(0.157)	5(0.079)	1(0.051)	2(0.051)	3(0.027)
5,4	5(0.097)	3(0.069)	1(0.049)	6(0.047)	2(0.03)
5,4,5	1(0.072)	—	—	—	—
—	2(0.261)	4(0.131)	5(0.065)	1(0.033)	3(0.001)
2	4(0.157)	5(0.079)	3(0.051)	1(0.039)	2(0.026)
2,4	5(0.098)	1(0.049)	3(0.033)	—	—
2,4,5	1(0.072)	3(0.016)	—	—	—
—	—	—	—	—	—

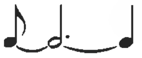
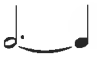
Table 8.3: Expectancies obtained for the melody *Boys and Girls Come Out to Play.* In this table, the first item in each cell represents the expected note, while the second item represents the actual expectation level for that note. In every case, the expectation produced matched what was due to occur next.

8.4.3 Roberts' rhythm segmentation experiments

Roberts [133, Chapter 6] described experiments that showed how a set of ten diverse rhythms could be processed using a time-driven SONNET configuration (see Appendix A). At $\mathcal{F}_1$, the content of each rhythm is represented using the IOI representation described in Chapter 7. Each rhythm was presented ten times, with the chunk-outs on the final presentation constituting the performed segmentation.

Label	Rhythmic phrase	Epoch
$\mathcal{A}$		1
$\mathcal{B}$		1
$\mathcal{C}$		1
$\mathcal{D}$		4
$\mathcal{E}$		4
$\mathcal{F}$		5
$\mathcal{G}$		7

(a)

$\mathcal{B}$	$\mathcal{D}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{G}$	$\mathcal{A}$	
$\mathcal{B}$	$\mathcal{D}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{E}$	$\mathcal{F}$	

(b)

Table 8.4: The (a) learned patterns and (b) segmentation for the *Pink Panther* rhythm.

In our simulations, we were able to reproduce the segmentations reported by Roberts [133, Section 6.8.2]. For example, Table 8.4 shows the segmentation performed for the *Pink Panther* rhythm. As far as we are aware, this is the first time that Roberts' rhythm experiments have be reproduced.

8.5 Summary

This chapter pointed out a variety difficulties with the existing SONNET formulation and introduced modifications to address these difficulties. In particular, phrases repetitions can now be adequately processed and a multiplicative $I_i^{\times}$ formulation can now adequately measure arbitrary levels of mismatch between STM and LTM. These modifications were verified by reproducing previous SONNET experiments by Nigrin [114, Chapter 4], Page [118, Chapter 7] and Roberts [133, Chapter 6].

These modifications are necessary to enable SONNET to be used and further extended in a more versatile manner. The coming chapters will show how SONNET can be extended to represent a variety of melodic features and how these features can be processed in parallel using SONNET-MAP.

Chapter 9

Representing Melodies in SONNET

9.1 Introduction

This chapter introduces a shared event field, labelled $\mathcal{F}_0$, that temporarily records the notes of a melody as they occur. Methods are presented that show how certain characteristic features of a melody, discussed in Chapter 2 (IOIs, IPCs, pitch-intervals and melodic contours), can be derived from $\mathcal{F}_0$ and represented at $\mathcal{F}_1$ where they are further encoded in LTM. A generalized $\mathcal{F}_1$ equation is introduced that can be used to represent each of these features at $\mathcal{F}_1$. This facilitates the use of a new method of representing pitch-intervals and melodic contours. In particular, the use of pitch-intervals for melody segmentation will prove beneficial (see Chapter 12). Furthermore, methods of combining different melodic features at $\mathcal{F}_1$, which facilitate improvements in melody processing (see Chapter 11), are presented. Although a single $\mathcal{F}_1$ field can represent two melodic features, this chapter argues that such a configuration is unfavourable. Instead, the use of a separate $\mathcal{F}_1$ field to process each feature independently is favoured. Finally, the manner by which SONNET stores melodies in LTM is discussed. Although the storing of melodies using connection strengths is not new with respect to ANNs, it is a novel approach with respect to melody retrieval.

Many illustrated examples will be used to explain the rhythm and pitch representations in this chapter. Therefore, in order to have a common point of reference, all examples will be based on the melodic fragment shown in Figure 9.1.

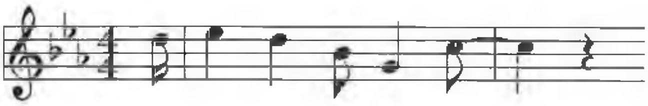


Figure 9.1: The first six notes of the melody *The Long and Winding Road.*

9.2 Using an event field at $\mathcal{F}_0$ to record note occurrences

In order to receive melodies and temporarily store the notes that comprise these melodies, an event field at $\mathcal{F}_0$ is introduced. From $\mathcal{F}_0$, a myriad of information is available to derive IOIs, IPCs, pitch-intervals and melodic contours, each of which can then be represented at the $\mathcal{F}_1$ field of a SONNET module. The manner by which these features can be derived from $\mathcal{F}_0$ and represented at $\mathcal{F}_1$ is described in Sections 9.5 and 9.6. Furthermore, Section 9.7.2 shows how $\mathcal{F}_0$ can be shared by any number of $\mathcal{F}_1$ fields, each processing a separate feature. For now however, the remainder of this section presents the details of $\mathcal{F}_0$'s architecture and operation.

9.2.1 Each $\mathcal{F}_0$ cell represents a categorical pitch-height

Complex tones, despite consisting of several harmonics, are generally associated with a single pitch; that of the fundamental frequency. Therefore, the pitch sensation of both simple and complex tones can be considered as perceptual units known as a pitch-heights (see Section 2.3). Furthermore, an infinite number of pitches exist in the pitch-height continuum. However, for representational purposes, a finite number of cells need to approximate this infinite range. $\mathcal{F}_0$ could consist of a single cell for each JND spanning the entire range of usable pitches, which is approximately eight octaves. According to Sano and Jenkins [140], approximately 1,852 cells would be required to achieve this. For practical purposes however, this thesis uses buckets at the semitone foci instead, with each octave containing twelve chromatic pitch categories. Furthermore, analogous to the layout of a piano keyboard, each semitone is tuned with respect to *middle C* at *concert pitch.* Therefore, a *categorical pitch-height* (CPH) representation is used at $\mathcal{F}_0$ to model pitches rather than an absolute pitch-height representation. Consequently, $\mathcal{F}_0$ is divided into a field containing 88 cells, each responding to a particular CPH (see Figure 9.2): spanning low A to high C. In line with Sano and Jenkins [140], an absolute pitch-height that is not represented by an $\mathcal{F}_0$ cell will be incorporated into the nearest CPH represented at $\mathcal{F}_0$. Therefore at $\mathcal{F}_0$, a melody may be defined as *a sequence of categorical pitch-heights that occur at particular points in time, each lasting for a particular length of time.*

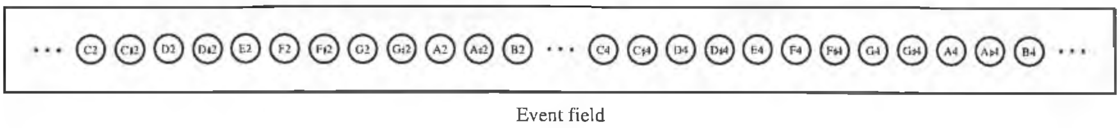


Figure 9.2: The $\mathcal{F}_0$ field consist of cells that represent CPHs spanning the entire audible range. To some degree, $\mathcal{F}_0$ is a very simple model of a tonotopically organized echoic memory. For more information, refer to Chapter 2.

9.2.2 Presenting melodies to $\mathcal{F}_0$

Each note of a melody is represented by a duple containing a CPH and its corresponding duration; {*CPH, duration*}. The labels applied to each CPH are its tone chroma and octave number. Rests and beats are special cases and are represented by the duples {*R, duration*} and {*B, duration*} respectively. Beats are useful for simulating recency effects (see Chapter 12) and are also used by Page [118, Chapters 6 and 7]. The tonal and metrical structure of each melody is also provided. A melody's key is required in order to represent pitch sequences using IPCs at $\mathcal{F}_1$, while the tactus-span of a melody is provided in order to represent rhythms using IOIs at $\mathcal{F}_1$. In effect, melodic input is represented in much the same way as a musical score is. Table 9.1 shows the structure of a melodic input file representing the fragment from Figure 9.1.

Tactus-span	0.89 seconds
Key	E♭ major
D4	0.25
E♭4	1.0
D4	1.0
B♭4	0.5
G4	1.0
C4	1.5
R	1.0

Table 9.1: A melody is presented to $\mathcal{F}_0$ as a sequence of {*CPH, duration*} duples. A melody's tactus-span and key are also supplied. Note also that the durations are given in terms of the tactus-span. For example, the duration of the first note is $\frac{1}{4}$ of a tactus-span, which is approximately equal to 0.22 seconds (0.25×0.89). By specifying each duration in terms of the tactus-span, the tempo of a melody can be changed by altering this single quantity rather that by altering each individual duration.

Initially, every $\mathcal{F}_0$ cell is set to zero. Then, each duple is presented in sequence to $\mathcal{F}_0$. The tonotopic organization of $\mathcal{F}_0$ solicits the activation of the appropriate cell in response to the CPH specified by each duple. This cell activates to unity and remains active for the duration specified. When the specified time duration has elapsed the cell is reset back to zero. When a rest is presented, $\mathcal{F}_0$ will remain inactive for the duration of the rest. Although the activation of each cell is only fleeting, all of the required information is available to derive and represent IOIs, IPCs, pitch-intervals and melodic contours at $\mathcal{F}_1$ (see Sections 9.5 and 9.6).

9.2.3 Assumptions regarding melodic input to $\mathcal{F}_0$

Pitch and rhythm

In addition to raw acoustic signals, melody sources include various file formats such as WAV, MP3, WMA and MIDI. For each of these sources, techniques exist that can extract pitch and rhythm information with varying degrees of success. The information required by this thesis is the CPH and duration of each note occurrence of a melody. However, the extraction of such information can encompass an entire research field in itself, and therefore is beyond the scope of this thesis. Consequently, the preliminary analysis of a melodic source is assumed to have already been performed by a subsystem capable of extracting a melody's CPHs and their corresponding durations.

Although ANN research exists which investigates the low level perception of pitch [64, 140, 155], most ANNs that perform high-level musical tasks, such as learning, recall, composition or tonal analysis, work with a similarly high level melody representation and make the same assumption regarding their derivation from a melody source (including previous SONNET-based research and, indeed, CBR research) [5, 6, 7, 56, 57, 59, 73, 78, 87, 105, 107, 119, 141, 159]. Decoupling abstract notions of pitch and rhythm from their low level representation in a melody source enables a research project to remain independent of the methods available to extract this information from an ever-evolving and ever-increasing range of melody sources. Furthermore, from a design and implementation point of view, adaptors for note extraction engines written for a particular melody source can be developed and plugged in as the demands of a specific application domain require. We consider this advantageous.

Tonal and metrical structure

The tonal and metrical structure of a melody is a core aspect of music cognition. Recent research has utilized ANNs in studying tonality [6, 7, 59, 81, 141] and tempo [133]. Although a melody's metrical and tonal structure can be inferred from $\mathcal{F}_0$, this inference process is beyond the scope of this thesis. Therefore, in addition to assuming a mechanism for deriving categorical pitch-heights and their durations, this thesis also presupposes the existence of mechanisms for determining a melody's key and tactus-span.

9.2.4 Comparisons with Page's event module

Page [118, pg. 247] uses an *event module* at the root of each SONNET hierarchy. This module is similar to the $\mathcal{F}_0$ field presented here in that this event module consists of an array of cells, each of which responds to the occurrence of a note onset. However, $\mathcal{F}_0$ differs from Page's event module in a variety of ways.

1. Each $\mathcal{F}_0$ cell represents a CPH, whereas each cell of Page's event module represents a diatonic scale note (see Section 7.2).

2. The cells of Page's event module are latch cells that activate for a predetermined length of time in response to a note onset. Whereas, each $\mathcal{F}_0$ cell activates and remains active for the entire duration of each note.

3. Page's event module is intended to model the communication of classifications between adjacent SONNET modules within a hierarchy. Consequently, there is a one-to-one correspondence between each latch cell at the event module and each input cell at $\mathcal{F}_1$. In effect, there is little difference between activating $\mathcal{F}_1$ cells via the event module and activating $\mathcal{F}_1$ cells directly when a note is presented. In comparison, the activation of each $\mathcal{F}_0$ cell is merely used to temporarily record a note occurrence, from which the characteristic features of a melody can be derived. There is no one-to-one correspondence between $\mathcal{F}_0$ and $\mathcal{F}_1$. Instead, computational maps (or gating mechanisms) are used to facilitate communication. Furthermore, as Section 9.7.2 shows, $\mathcal{F}_0$ can be shared by separate, parallel SONNET modules, each of which processes a specific melodic feature.

9.3 Methods of representing information at $\mathcal{F}_1$

Using the current SONNET architecture, there are two methods of representing information at $\mathcal{F}_1$. Each method stores order information in the usual manner — using a decreasing activity pattern (5.3.1). The methods differ however, in terms of how individual categories are represented. Both approaches have been implicitly described in previous chapters. However, this chapter explicitly identifies each method and discusses the various issues involved in making a suitable choice between them.

9.3.1 Using a distinct sibling group per category

This scheme uses a distinct sibling *group per category* (GPC), utilizing a *multiple sibling group* (MSG) architecture at $\mathcal{F}_1$. In response to the presentation of an input category, any sibling within the appropriate sibling group fires. This scheme, which is arguably the most obvious, is used by the majority of ANNs. This includes the work by Nigrin [114], Page [118] and Roberts [133, Chapter 4] to represent letter sequences, scale-notes and quantized IOIs respectively (see Section 8.4). This scheme is particularly well suited to categories that are completely defined by a single event (IPCs for example).

9.3.2 Using cell activation ratios

This scheme utilizes *cell activation ratios* (CARs) to represent categories at $\mathcal{F}_1$. That is, a category is represented by the ratio of the activation levels between two input assemblies that fired in response to consecutive event onsets. Consequently, only a single sibling group which represents event onsets is required to store information using CARs. This method is particularly well suited to categories that are defined in terms of the interval between two events. For example, IOI's (see

Chapter 7), pitch-intervals and melodic contours. Additionally, when using CARs to represent categories, the last event onset of one pattern may actually be the first event onset of a different pattern. Consequently, the last onset of a pattern should not be reset during the chunking-out process (K_{chunk} = false) when CARs are used to represent categories (see Section 7.3.4).

A combination of the CAR and GPC schemes may be utilized to represent two melodic features using a single $\mathcal{F}_1$ field. However, Section 9.7.1 discourages this approach. Therefore, unless stated otherwise, a *single sibling group* (SSG) architecture should be assumed when the CAR representational scheme is utilized.

9.3.3 Comparing $\mathcal{F}_1$ representational schemes

The choice of representational scheme at $\mathcal{F}_1$ will depend on the quality of the information that is available and the target application. The advantages and disadvantages of each scheme need careful consideration. This section discusses the major issues to be investigated in making a suitable choice of representational scheme (note that some of the points raised here relate to those discussed by Roberts [133, Section 5.2] with respect to the benefits of time-driven networks over event-driven networks, paying particular attention to IOIs).

Pre-processing. Generally, when dealing with a melody source, the GPC scheme requires more pre-processing than the CAR scheme. For GPC schemes, each musical event (whether an IOI, an IPC, a pitch-interval or a contour) needs to be classified or quantized to a category that exists at $\mathcal{F}_1$. Then, the event's category needs to be mapped to a particular sibling within the appropriate sibling group. Using CARs, the level of processing depends on the musical feature to be represented at $\mathcal{F}_1$. For example, pitch intervals still require knowledge about the absolute pitch-heights that bracket the interval. However, no mapping to any particular sibling group is required. In response to a note onset, the only requirement is the activation of any inactive input assembly. For IOIs, other than identifying the onset of each note, no pre-processing is required at all nor is any mapping to any particular sibling group.

Prior knowledge of input categories. For GPC schemes, each possible category needs to be known in advance. Otherwise, the mapping of specific input events to specific sibling groups cannot occur. In contrast, using CARs, an arbitrary number of categories can be represented at $\mathcal{F}_1$. Furthermore, these categories need not be known in advance.

Extensibility. For GPC schemes, prior knowledge of input categories is a requirement. However, it is not always possible nor desirable to know every possible input category in advance. Tying each input category to a specific sibling group is restrictive because only input sequences containing known categories may be processed. This limits the extensibility and adaptability of architectures that use GPC representational schemes. For example, expressive timing cannot be exploited using a predetermined set of IOI's. Similarly for pitch categories, unless categories have been specified over a broad range, the music of other cultures may not be adequately represented. In contrast, using CARs, expressive timing can easily be represented and processed. Indeed, sequences containing any combination of pitches can be easily represented because intervals are not tied to any particular musical culture, such as Indian ragas and Western popular music. Furthermore, a singers inability to perfectly reproduce the pitches of a melody may be represented, which is important for query-by-humming retrieval systems.

The number of $\mathcal{F}_1$ input assemblies. When processing input sequences, the size of $\mathcal{F}_1$ needs to be at least as large as the maximum number of events that can be concurrently stored in STM. For example, using the GPC scheme, 84 input assemblies are required to represent the thirteen pitch-interval categories of Western tonal music (including the unison), assuming a maximum of seven items can be concurrently stored at $\mathcal{F}_1$ ($N_1^g = 12$ and $N_1^s = 7$ giving $N_1^g \times N_1^s = 84$). In contrast, a CAR schemes only requires eight input assemblies ($N_1^g = 1$ and $N_1^s = 8$ giving $N_1^g \times N_1^s = 8$). Remember, eights note onsets will represent seven pitch-intervals. This is a practical advantage that comes into its own when scalability is an issue.

Segmentation. The segmentations formed may depend greatly upon the representational scheme used. For example, large pitch-intervals may be more difficult to learn than small pitch-intervals using CARs. This is particularly true if the excitatory weight vector is initialized in a manner that biases learning towards forming phrase categories with relatively small intervals (modelling pitch proximity, which would be desirable when trying to model pitch perception). Whereas, using a GPC scheme, the size of the interval has no influence on the segmentation process whatsoever without additional network structure [88], where only sequential regularities are discovered.

Recognition. The ability of $\mathcal{F}_2$ cell assemblies to recognize learned phrase categories across $\mathcal{F}_1$ is of great importance; particularly for applying SONNET-based networks to the problem of CBR of melodies. Recognition is considerably easier using GPC schemes when input environments contain low noise levels. However, recognition is not robust for GPC schemes when noise levels are relatively high. For example, χ_i will not recognize a phrase (or query) containing an incorrect pitch, even if the pitch is out by only a single semitone. χ_i makes no distinction between an incorrect pitch and an absent pitch. This is because an incorrect pitch will activate an input assembly that is not a member of χ_i's receptive field T_i. In contrast, although the recognition of phrases is more difficult using CARs, recognition degrades more gracefully as noise levels increase. This is because, although a pitch may be incorrect, its occurrence is still acknowledged. Furthermore, a measure of by how much it differs from the expected pitch is built into the recognition process by virtue of the comparison performed between normalized excitatory weights and their corresponding normalized input during the $I_{ji}^{\times}$ computation. In effect, the similarity between the pitches at $\mathcal{F}_1$ and those of a previously learned phrase category is directly measured. This turns out to be a considerable advantage and is illustrated in Chapter 13.

Each of the representational schemes discussed in this section can be implemented using the equation that is introduced in the next section. Following that, it will be shown how the characteristic features of rhythm and pitch can be represented at $\mathcal{F}_1$ using this formula.

9.4 Modifications to the $\mathcal{F}_1$ dynamics

This section introduces generalized $\mathcal{F}_1$ dynamics. This facilitates the use of a new method of representing pitch-intervals and melodic contours using CARs, which cannot be easily achieved using previous $\mathcal{F}_1$ dynamics. In particular, the use of pitch-intervals for melody segmentation will prove beneficial (see Chapter 12).

9.4.1 Generalizing and simplifying the input cell dynamics

Equation 5.1 models the dynamics of an on-centre off-surround circuit. This circuit can be used to store event occurrences in the order in which they occurred. Section 5.4.3 presented a simplification of this equation, which was introduced by Roberts for networks that operate in event-driven mode. Roberts further introduced Equation 7.12, which is used to represent IOIs when SONNET is operating in time-driven mode. Although this equation is simplified with respect to Equation 5.1, a further simplification is introduced in this section, which generalizes across all SONNET processing modes (whether using an event-driven or a time-driven configuration) and across any possible $\mathcal{F}_1$ encoding schemes (see Sections 9.5 and 9.6). In general, when an event (e_i) occurs:

1. Every input cell is scaled using the following equation:

$$s_{mj}(i) = s_{mj}(i-1)\omega^{\lambda} \tag{9.1}$$

2. Any inactive input assembly within e_i's corresponding sibling group is set to μ.

i represents the current event number and λ is a value dependent on the type of representation desired. A value of $\lambda = 1$ indicates an isochronously presented input pattern. Whereas a value of $\lambda = 0.5$ could represent an IOI of 300ms assuming $T_{span} = 600$ms. Later in this chapter we show how λ can be used to allow pitch-intervals and melodic contours to be represented at $\mathcal{F}_1$.

9.4.2 $\mathcal{F}_1$ overload based on activation time

Section 5.3.1 described how input assemblies are reset to limit the depth and capacity of STM. This reset mechanism is based on total field activation (g_s) for networks operating in event-driven mode or individual input assembly activations (s_{mj}) for networks operating in time-driven mode. Therefore, activation levels at $\mathcal{F}_1$ provide the information required to control $\mathcal{F}_1$ resets. In event-driven mode, with $\mu = 1$, $\omega = 2$ and $T_{attn} = 0.2$ seconds, an activation level of 64 at γ_{mj} indicates this assembly has been active for six event onsets ($\omega^6 = 2^6 = 64$). Therefore, γ_{mj} has

e	Note value	IOI	Scale	Total	s_j
1	—	—	—	—	1
2	𝅘𝅥𝅯	$\frac{1}{4}$	$\omega^{\frac{1}{4}}$	$\frac{1}{4}$	1.32
3	♩	1	ω^{1}	$1\frac{1}{4}$	3.95
4	♩	1	ω^{1}	$2\frac{1}{4}$	11.85
5	♪	$\frac{1}{2}$	$\omega^{\frac{1}{2}}$	$2\frac{3}{4}$	20.52
6	♩	1	ω^{1}	$3\frac{3}{4}$	61.55
7	𝅗𝅥 + ♪	$2\frac{1}{2}$	$\omega^{2\frac{1}{2}}$	$6\frac{1}{4}$	959.42

Table 9.2: This table shows the activation level of a single input cell s_j after the events of Figure 9.1 have been presented. In this example, $\mu = 1$, $\omega = 3$ and $K_{12} = \omega^6 = 729$.

been active for 1.2 seconds. Similarly, in time-driven mode, with $\mu = 1$, $\omega = 3$ and $T_{span} = 0.6$ seconds, an activation level of 729 at γ_j indicates this assembly has been active for six tactus spans ($\omega^6 = 3^6 = 729$). Therefore, γ_j has been active for 3.6 seconds. From these examples, it can be seen that there is a correlation between an input assembly's activation level and the length of time it has been active for. Consequently, $\mathcal{F}_1$ overloads are related to the length of time input assemblies have been active for.

Basing $\mathcal{F}_1$ overloads on input assembly activation levels leads to difficulties when generalizing and simplifying the $\mathcal{F}_1$ dynamics over all possible representational schemes. In particular, the use of activation levels cannot generally be used to determine $\mathcal{F}_1$ overloads for representations based on CARs. To provide an example of this, Table 9.2 shows the activation level of s_j after the six notes from Figure 9.1 have occurred. In this example, $\mathcal{F}_1$ represents IOIs using CARs. Utilizing the generalized and simplified input cell dynamics, s_j will have exceeded K_{12} at the instant the seventh event onset has occurred. However, at this point in time, s_j will have been active for $6\frac{1}{4}$ tactus-spans. This is incorrect, because according to K_{12}, the input cell should have been active for a maximum of six tactus-spans before being reset. This problem occurs because, using the simplified $\mathcal{F}_1$ dynamics, the activation level of s_j is *not* directly correlated with the length of time it has been active for. Therefore, it is unclear how the overload threshold should be chosen. This problem is even more pronounced when representing pitch-intervals using CARs.

A capacity based overload scheme could be utilized to limit the depth and capacity of STM. For example, the total number of active input cells could be restricted to a maximum of seven. Suppose seven input assemblies are active when an eight event occurs. This situation would lead to $\mathcal{F}_1$ overload and subsequent input cell resets. Although this solution would work well before cell assemblies have learned any patterns, problems occur for networks that have already learned patterns. When patterns have been learned, it is more likely that $\mathcal{F}_1$ cells will be reset due to chunking-out rather than by $\mathcal{F}_1$ overloads. Then, onsets that do not belong to any learned pattern may never be reset because the field never has more than seven active input assemblies. This is clearly undesirable.

Instead, a simple solution to resolve these issues is to base $\mathcal{F}_1$ overload on cell *activation times*[1] rather than *activation levels*. Therefore, regardless of the specifics of a particular representational scheme at $\mathcal{F}_1$, cells will reset consistently. Therefore, in the example above, s_j will be reset between the occurrence of the 6^{th} and 7^{th} events.

In the case of an event-driven network which uses a GPC representational scheme, $\mathcal{F}_1$ overload and reset occurs as follows. When any input assembly has been active for a time period of K_{12}, at most four input assemblies which have been active for a time period of K_{11} are considerred for reset. Of this set, the three input assemblies that represent the oldest input events are reset. Additionally, the input assembly that represents the fourth oldest input event may also be reset. There is a 50% chance of this happening.

For both event-driven and time-driven SONNET configurations, this mechanism provides the exact same functionality as before. However, this mechanism is more flexible and extensible to other encoding schemes.

9.5 Representing rhythm at $\mathcal{F}_1$

Roberts' work on interpreting rhythmic structures showed how IOIs can be represented at $\mathcal{F}_1$. Two possibilities exist to achieve this:

1. A GPC scheme can be used to represent IOIs at $\mathcal{F}_1$ [133, Chapter 4]. IOIs from a melody source can be quantized into standard notation for musical time categories (for example: quavers, crotchets and minims). Then, a separate $\mathcal{F}_1$ sibling group will respond to each distinct category as a rhythm is presented (see Figure 9.3).

2. CARs can be used to represent IOIs at $\mathcal{F}_1$ [133, Chapter 5]. Note onsets are represented by a single sibling group. The activations levels of input assemblies, which have responded to note onsets, continuously change. Consequently, the ratio between activity levels associated with consecutive onsets will represent IOIs (see Figure 9.4).

The simplified $\mathcal{F}_1$ dynamics introduced in Section 9.4.1 can be utilized to enable IOIs to be represented at $\mathcal{F}_1$ using CARs. This entails calculating the quantity λ, which in this circumstance represents the time interval between consecutive note onsets in terms of the tactus-span. To this end, a timing cell labelled t_{IOI}, is introduced at $\mathcal{F}_1$. t_{IOI} represents the time interval that has elapsed since the occurrence of the most recent note onset. t_{IOI} is similar to the t_{ni} quantity used in the time dependence function τ_{ni} (described in Section 7.3.6). When an event occurs, the current value of t_{IOI} is used to calculate the term λ before being reset back to zero. λ is calculated as follows:

$$\lambda = \frac{t_{IOI}}{T_{span}} \tag{9.2}$$

[1] The use of activation times is not new to SONNET; for example, the chunking-out delay (K_{14}) and the time dependence functions (τ_{ni}) are both based on activation times.

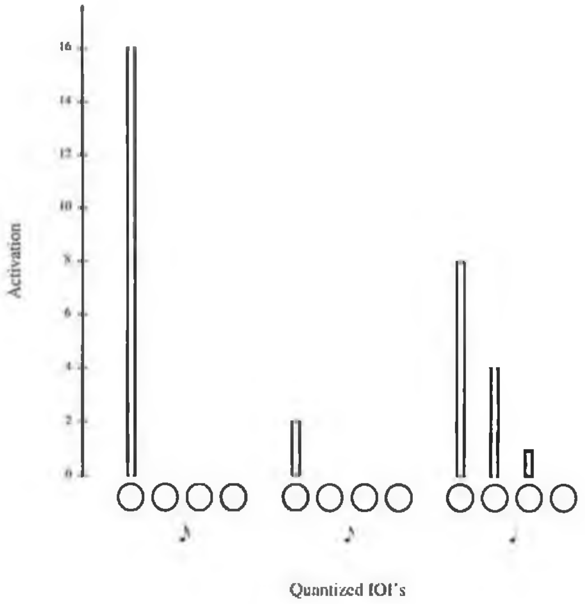


Figure 9.3: STM storage of Fragment 9.1 at $\mathcal{F}_1$ using a distinct sibling group for each quantized IOI.

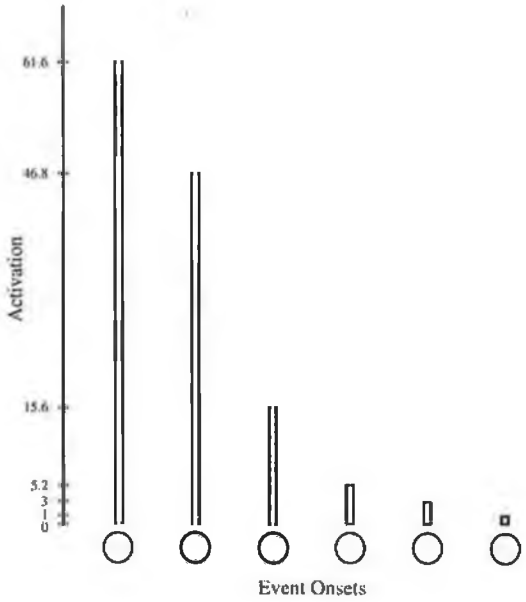


Figure 9.4: STM storage of Fragment 9.1 at $\mathcal{F}_1$ using CARs to represent IOIs.

In effect, t_{IOI} represents the actual IOI in seconds, whereas λ represents the IOI as a fraction of the tactus-span.

These simplified dynamics behave in exactly the same manner as Roberts' original specification. However, the input assembly activation levels are computed only once per note onset instead of on a continuous basis (that is, after each time step of T_{step}). This reduces computation time, which is at a premium when scaling-up to larger problems. Furthermore, since a time-based overload mechanism is now used at $\mathcal{F}_1$ (see Section 9.4.2), the resetting of $\mathcal{F}_1$ input assemblies will occur at the same points as specified by Roberts' original sliding window mechanism (see Section 7.3.2).

9.6 Representing pitch sequences at $\mathcal{F}_1$

Possible representations for pitch sequences in STM and LTM include: IPCs, pitch-intervals and melodic contours. The choice of representation to be used for pitch is a contentious issue and there has been a great deal of debate regarding which is the most appropriate representation to use for neural networks. This thesis argues that each of these representations exist and the question should not be one of choice between each possibility, but one of how these representations can complement each other. This point is particularly important for the work presented in Chapter 13, where the use of particular representation may depend on the quality of the input query. For now however, the remainder of this section will show how each of these representations can be encoded in working memory at $\mathcal{F}_1$.

9.6.1 IPC representation

The IPC representation can be derived in the following way. First, CPH is collapsed across all octaves. This reduces the pitch range to a single octave containing pitch-classes; thus accounting for octave equivalence. However, since absolute pitch levels within this collapsed octave are preserved, invariance under transposition is not accounted for. This problem is addressed by normalizing each sequence with reference to a tonal centre, or key. In the case of Western tonal music employing equal-tempered tuning, each pitch is represented by one of twelve chromatic pitch categories depending on a melody's tonal centre. These categories are labelled from $0 - 11$. For example, the *tonic* will always be represented by 0, regardless of its absolute pitch.

A neural gating mechanism for computing IPCs

Bharucha [6] describes how IPCs can be derived from a pitch-height representation using the MUSACT neural network. A similar mechanism is also described by Scarborough et al. [141]. The *gating mechanism* presented here is derived from MUSACT's gating mechanism and is used to determine the IPC of each note using two pieces of information; the CPH of each note and the melody's key, both of which are supplied as input. The gating mechanism consists of a pitch-class field containing a cell for each of the twelve chromatic pitch categories, a tonal centre field containing a cell for each possible key and a set of π cells that multiply both their input signals

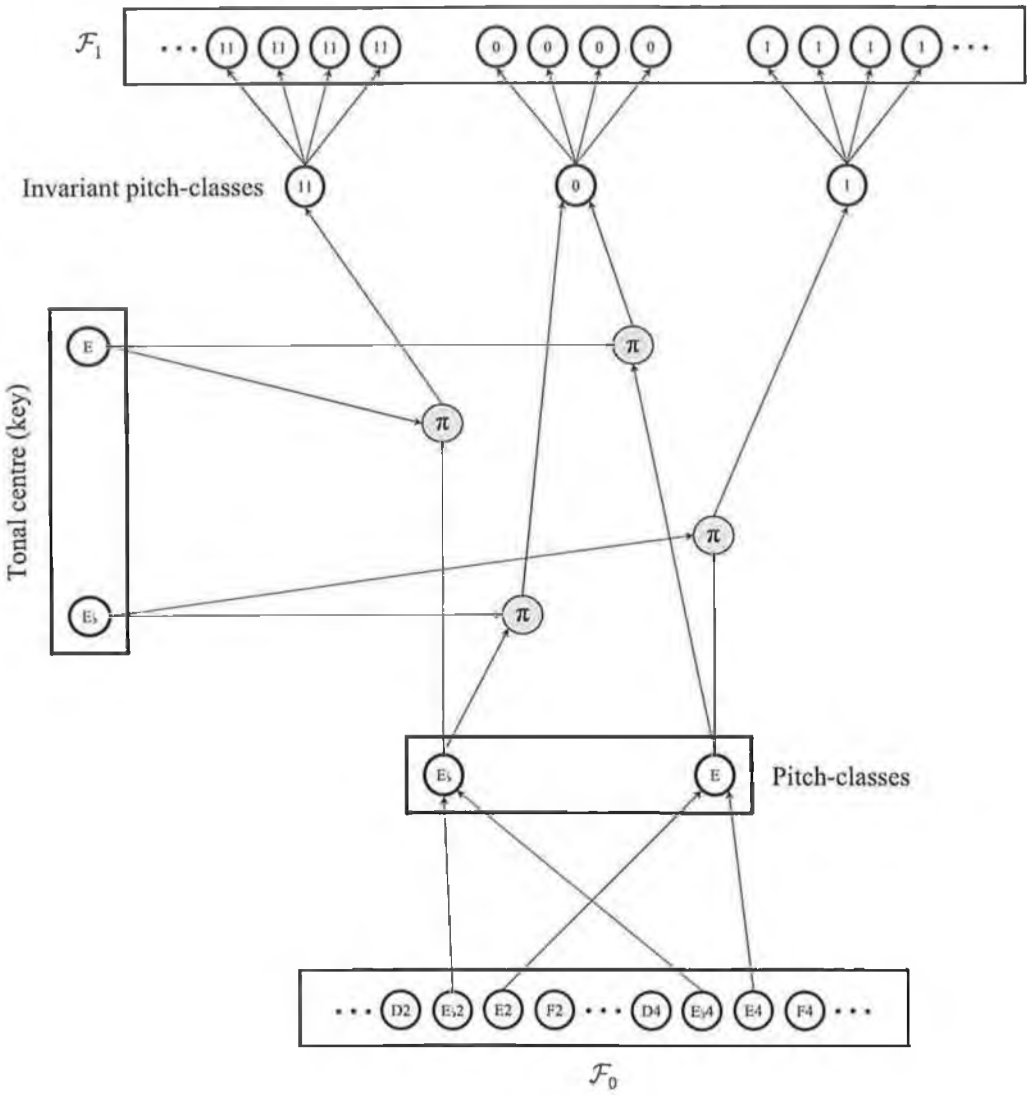


Figure 9.5: The neural gating mechanism used to compute IPCs from a CPH and a melody's key. For simplicity, only the keys of E and E♭ are illustrated.

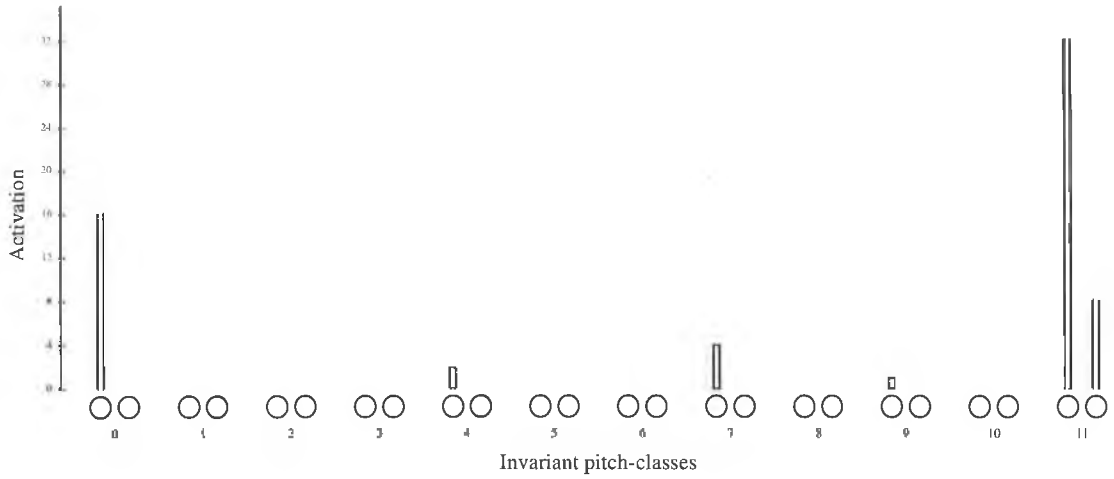


Figure 9.6: STM storage of fragment specified in Figure 9.1 using the invariant pitch-class representation with $\mu = 1$ and $\omega = 2$. The pitch-class sequence of this fragment is {D, E♭, D, B♭, G, C}. When normalized to the invariant pitch-class representation, this sequence is coded with respect to its tonal centre, which is E♭ major, giving an IPC sequence of {11, 0, 11, 7, 4, 9} illustrated above.

together — one from a pitch-class field and the other from a tonal centre field (see Figure 9.5). The CPHs are collapsed across all octaves by simply connecting all octave equivalent pitches at $\mathcal{F}_0$ to the same chromatic pitch category at the pitch-class field. For example, all of the CPHs at $\mathcal{F}_0$ that represent the pitch E (E1, E2, E3, E4, E5, E6, and E7) are connected to the same chromatic pitch cell at the pitch-class field.

On presentation, a melody's key activates the appropriate cell in the tonal centre field to unity. This cell remains active for the entire duration of the melody or until a key changes is indicated. The signal from this cell is then propagated to each π cell it is connected to. When a pitch-class cell is activated to unity, via $\mathcal{F}_0$, the signal from this cell is also propagated to every π cell it is connected to. When a tonal centre cell and a pitch-class cell are simultaneously active, the connectivity pattern of the gating mechanism will cause a single π cell to activate. This cell identifies the IPC, which in turns activates a sibling within the appropriate sibling group.

Representation IPCs at $\mathcal{F}_1$

Applying this to the input field of a SONNET module is straight forward. $\mathcal{F}_1$ consists of twelve sibling groups ($N_1^g = 12$), one for each invariant pitch category. When an IPC event occurs, one of the siblings within the appropriate group activates to an initial value of μ (due to the gating mechanism), while every other cell activity is scaled by a factor of ω. Note, since the relative activation between input assemblies is only required to preserve order information, λ remains constant at unity. In this way, the pitches and the order in which they occurred is stored in working memory in a octave and transpositional invariant manner (see Figure 9.6). Furthermore, since IPCs are defined by a solitary event (as opposed to being defined by two events, such as in interval representations), the only representational scheme that can be used is the GPC scheme.

IPCs have been used frequently by researchers in the field of *Music and Neural Networks* [60, 160]. For example, Page [118, 119] used an IPC representation consisting of only the diatonic categories or scale notes. (see Section 7.2). Page's work was primarily concerned with the phrase structures of very simple melodies that are based on nursery rhymes. Therefore, such a restricted category set was adequate. The IPC representation has not been used very much by CBR researchers, who predominantly use pitch-interval and contour representations. These representations are described next.

9.6.2 Pitch-interval representation

Using the pitch-interval representation, a pitch sequence is represented by the distance in pitch between consecutive notes. Again, in the case of Western tonal music employing equal-tempered tuning, thirteen interval categories need to be represented. One for each of the twelve intervals within an octave (including the octave interval) and one to represent sequences of identical pitches (see Table 9.3). It is important to point out that pitch-intervals must be computed in terms of pitch-heights to remove any ambiguities that may occur if they were computed using either the pitch-class or IPC representations. For example, using the IPC representation, the interval between the first two notes of Figure 9.1 would be computed as a *major* 7^{th} (or 11 semitones) when in fact the interval is actually a *minor* 2^{nd} (or a single semitone).

Octave at middle C	C	C♯	D	D♯	E	F	F♯	G	G♯	A	A♯	B	C
Interval in semitones	0	1	2	3	4	5	6	7	8	9	10	11	12

Table 9.3: Pitch-interval categories spanning an octave.

Using a neural difference gate for computing pitch-intervals

Pitch intervals which are presented to $\mathcal{F}_1$ are computed using a *difference gate* (see Figure 9.7), which is similar in operation to the gating mechanism used to compute IPCs. The difference gate consists of the $\mathcal{F}_0$ field, another identical field of CPHs labelled $\mathcal{F}_0'$ and a set of π cells. When a note onset is presented to $\mathcal{F}_0$, the current activity of this field is transferred to $\mathcal{F}_0'$ using a shift mechanism before $\mathcal{F}_0$ is reset. Then, the appropriate $\mathcal{F}_0$ cell activates in response to a new note onset in the usual way. The active cells at $\mathcal{F}_0$ and $\mathcal{F}_0'$ propagate their activity to every π cell. Due to the connectivity pattern of the difference gate, only a single difference cell will receive two inputs. This cell identifies the pitch-interval category, and is utilized differently depending upon the representation scheme used — a GPC scheme or a CAR scheme. Both possibilities are discussed next.

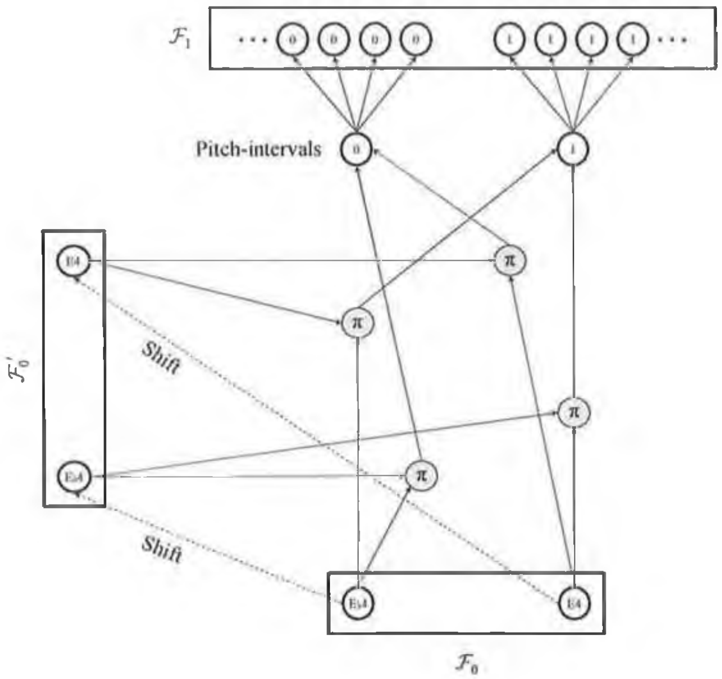


Figure 9.7: Pitch intervals are computed using a difference mechanism, which is similar in operation to that presented in Figure 9.5 for deriving IPCs. When new items are presented, a shift mechanism copies the current activity pattern across from $\mathcal{F}_0$ to $\mathcal{F}_0'$. This is similar to a tapped delay-line memory or a shift register. For simplicity, only the notes E and E♭ are illustrated.

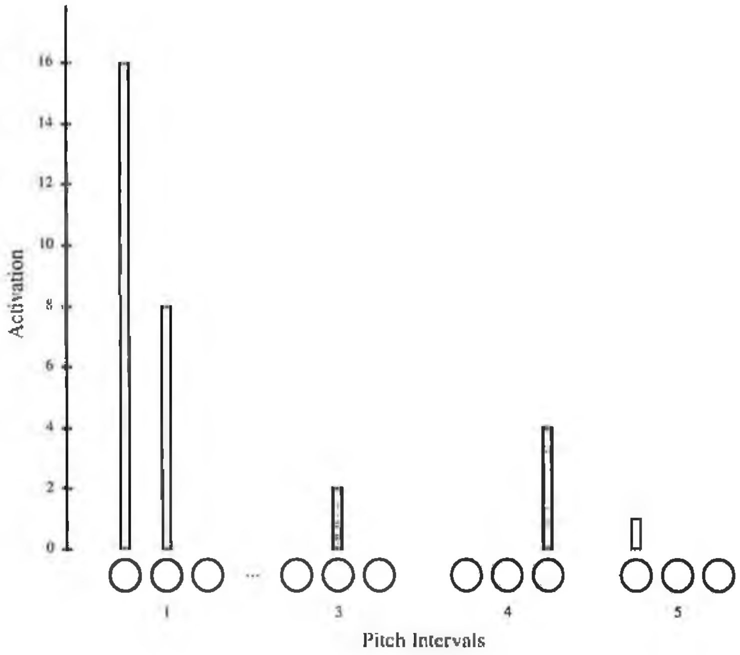


Figure 9.8: STM storage of Fragment 9.1 using the pitch-interval representation. In music theoretic terms, the sequence consists of {minor 2^{nd}, minor 2^{nd}, major 3^{rd}, minor 3^{rd}, perfect 4^{th}}

Representing pitch-intervals at $\mathcal{F}_1$ using the GPC scheme

Pitch-intervals can be represented in working memory using the GPC scheme. In other words, every sibling group at $\mathcal{F}_1$ will represent one of the thirteen pitch-interval categories described above. In this circumstance, the difference gate operates in exactly the same way as the IPC gating mechanism. That is, each difference cell multiplies its input, leaving a single π cell active. This cell identifies the interval, which in turn activates an input assembly within the appropriate sibling group (see Figure 9.8).

This scheme cannot directly measure the similarity between pitch-intervals stored in STM and pitch-intervals stored in LTM. This is problematic for input environments containing noisy input which leads to problems with CBR. A new mechanism, which addresses this problem, is introduced next.

Representing pitch-intervals at $\mathcal{F}_1$ using CARs

An alternative approach for represented pitch-intervals at $\mathcal{F}_1$ is introduced here. A single sibling group, which represents pitch onsets, is utilized. Then, the pitch-interval between successive notes in a melody is represented using CARs. The method described here is analogous to Roberts IOI representation, but novel with respect to pitch sequences.

A pitch-interval, and its associated IOI, are braced by the same note onsets. In fact, the only difference between this representation and the IOI representation is the calculation of the term λ in their respective $\mathcal{F}_1$ activation equations. For the IOI representation, λ measures the distance in time (with respect to T_{span}) between consecutive note onsets. Whereas, for pitch-intervals, λ measures the distance in pitch between consecutive note onsets (in semitones). With some slight adjustments, the difference gate can be used to compute λ. The difference gate's π_i cells operate as before in order to identify the pitch-interval between consecutive note onsets. An additional cell representing λ is connected to every π_i cell. Each connection has a fixed weight (PI_i) that represents the size of the pitch-interval identified by the π_i. In total, 88 distinct pitch-interval categories exist (including the unison), representing every pitch-interval within the 88 possible CPH's at $\mathcal{F}_1$. λ is calculated as follows:

$$\lambda = \frac{\sum_i [\pi_i(PI_i + 1)]}{K_{intervals}} \tag{9.3}$$

where $K_{intervals}$ is a quantity analogous the tactus-span for the IOI representation. For the segmentation experiments presented in Chapter 12, $K_{intervals} = 8$, which is the number of semitones spanning the interval of a *perfect* 5^{th}. For the retrieval experiments presented in Chapter 13, $K_{intervals} = 13$, which is the number of semitones spanning the interval of an *octave*.

This scheme is more advantageous than the IPC representation because less preprocessing is required (remember for IPCs, the tonal centre of the melody is required). Furthermore, the similarity between the intervals represented at $\mathcal{F}_1$ and those stored in LTM can be directly measured. Both of these advantages are beneficial for the CBR experiments described in Chapter 13.

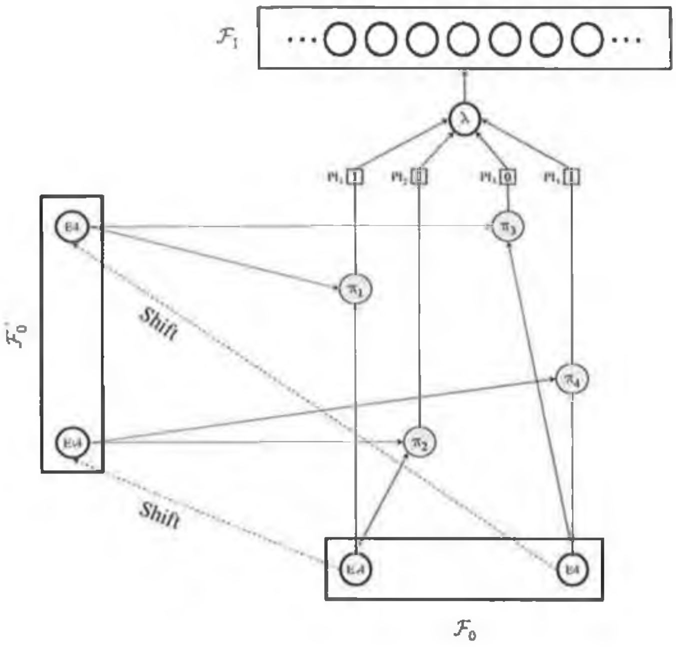


Figure 9.9: The difference mechanism for use with pitch-intervals represented by CARs. λ is calculated using the active π_i cell and the strength of the connection from π_i to λ. For simplicity, only the notes E and E♭ are illustrated.

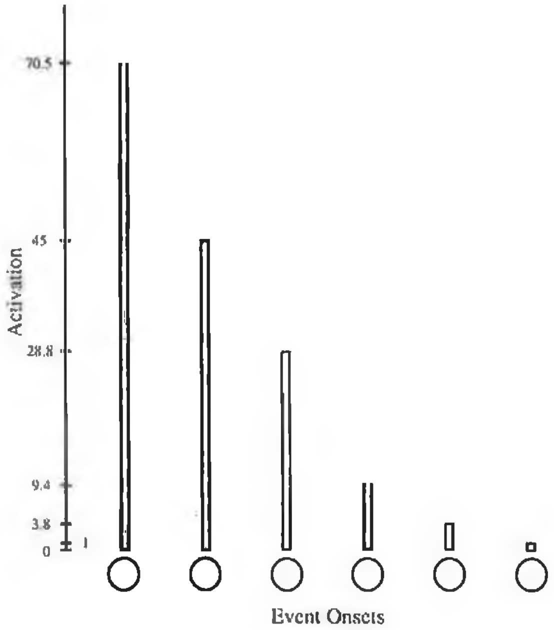


Figure 9.10: STM storage of Fragment 9.1 using CARs to represent pitch-intervals.

9.6.3 Contour representation

The IPC and pitch-interval representations described so far do not take into account the contour of a melody at $\mathcal{F}_1$. For example, the pitches G to C would activate their respective cells at $\mathcal{F}_1$. However, it would be impossible to tell whether the C in question is lower than the G (the interval of a perfect fifth) or the higher (the interval of a perfect fourth). Contour information is particularly useful for CBR when the input environment is noisy with poorly specified pitches. This is because pitches that are very poorly degraded can still be utilized to perform a retrieval, so long as the contour of the original melody has not been altered. Since contour information is important, it must be taken into consideration.

The contour representation can utilize the same techniques used by the pitch-interval representation. Three contour categories exist — S (same), U (up) and D (down), where U represents an increasing interval, D represents a decreasing interval and S represents a repeated pitch. Applying this to a GPC representational scheme in the usual way is simple. Encoding contours using CARs is also simple. By applying distance measures to each contour category (for example, S (1), U (2) and D (3)), contours can be processed in the same way as pitch-intervals that use CARs.

9.7 Combining different melodic dimensions

9.7.1 Using a single $\mathcal{F}_1$ field to represent multiple dimensions

Combining the techniques of the GPC and CAR representational schemes, it is possible for two dimensions of an input stimulus to be represented at a single $\mathcal{F}_1$ field. For example, pitch categories can be represented using sibling groups, while IOIs can be represented by CARs (see Figure 9.11). In fact, Lewis [88] brings both of these dimensions to bear on the problem of interpreting melodic structures. However, this approach suffers from a variety of problems. Firstly, it does not allow the recognition of a rhythm independently of its corresponding pitch sequence and vice versa. Roberts [133, Section 5.2] also considers the use of combining the CAR scheme with the GPC scheme and points out this problem too. Furthermore, this scheme is not extensible beyond two dimensions. Since almost all interesting input stimuli are multidimensional, SONNET needs the ability to represent *every* possible dimension of an input stimulus. These issues are a serious drawback and are clearly at odds with human melody recognition.

Secondly, this method is restrictive in terms of which representational schemes may be used together. For example, pitch-intervals and IOIs can not be represented at a single $\mathcal{F}_1$ field if the CAR representational scheme is employed. Because of these problems, the combination of multiple dimensions in this way is generally disadvantageous and is discouraged.

9.7.2 Using a separate $\mathcal{F}_1$ field for each dimension

This section proposes that each input dimension should be processed by separate but parallel $\mathcal{F}_1$ fields, with interactions and associations forming between these fields as appropriate (see Figure

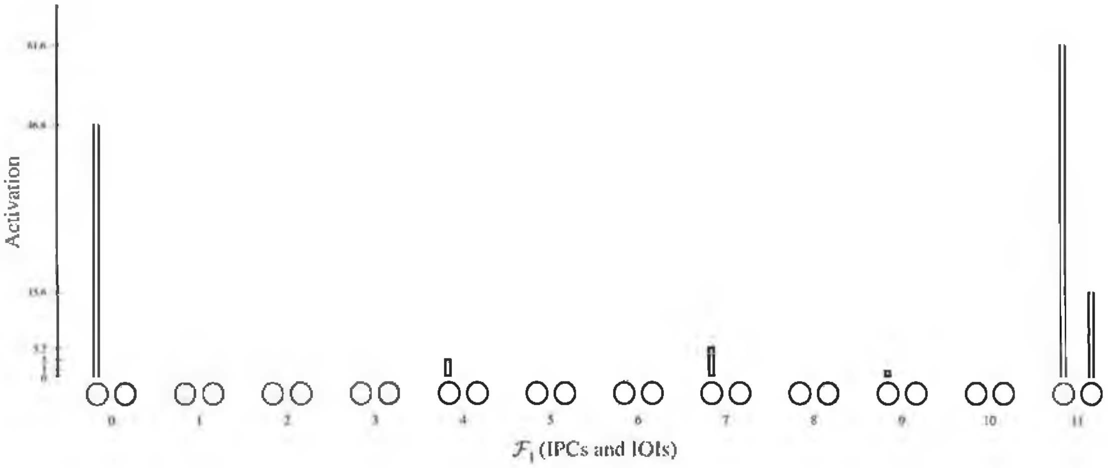


Figure 9.11: STM storage of Fragment 9.1 using IPCs and IOIs at a single $\mathcal{F}_1$ field.

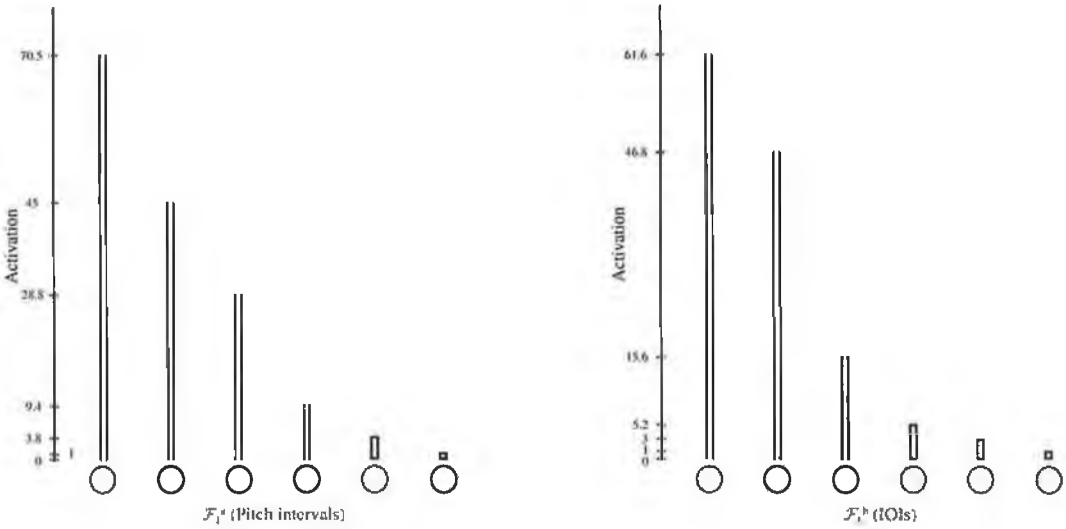


Figure 9.12: STM storage of Fragment 9.1 using pitch-intervals and IOIs at separate $\mathcal{F}_1$ field labelled $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ respectively.

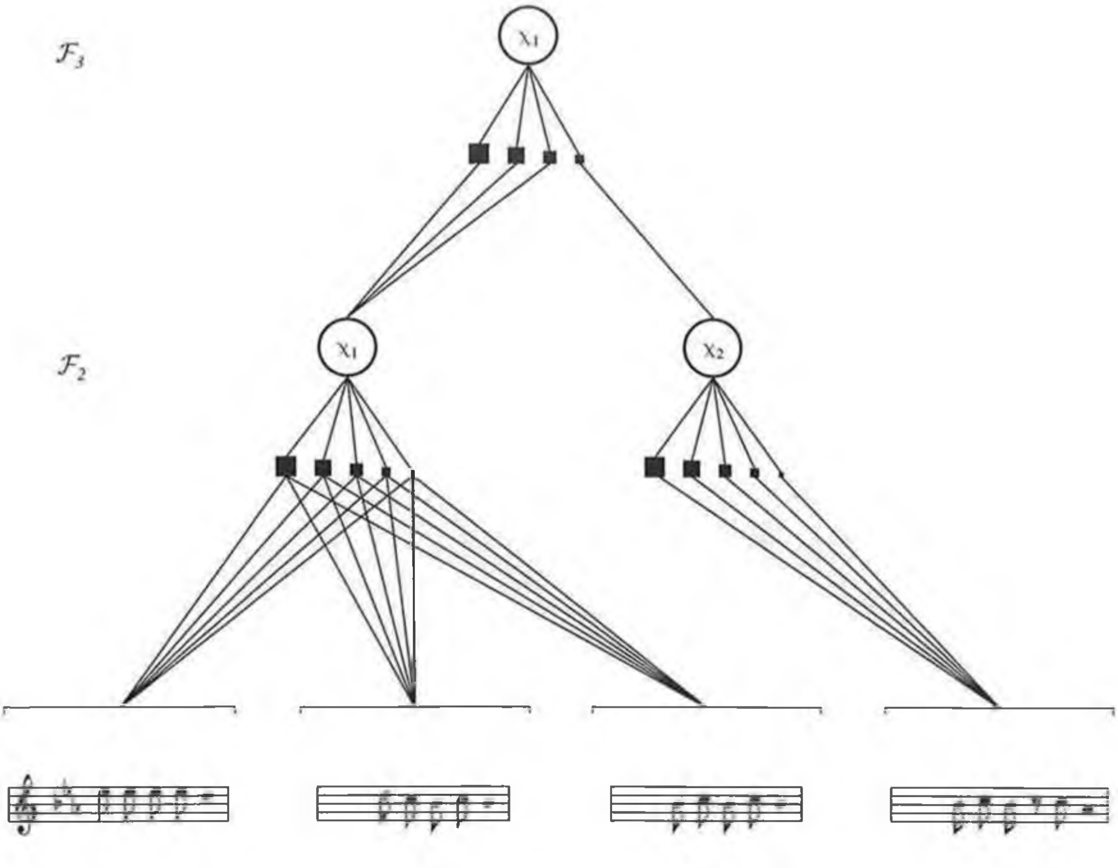


Figure 9.13: LTM representation of the IOI sequence for the first four segmented phrases of *Across the Universe*. Refer to Section 12.4 for the complete segmentation of this melody, which was performed by SONNET-MAP.

9.12). Indeed, this point has already been observed by Nigrin, Page and Roberts. In particular, Page [118, pg. 317] postulates the existence of parallel hierarchies suggesting that *"the interaction between parallel hierarchies, describing different aspects (pitches, rhythms etc.) of the same stimulus sequence needs to be investigated."*. However, the means by which this can be achieved for SONNET, and more generally, for temporal patterns, has undergone little investigation. Consequently, the next chapter introduces mechanisms by which this can be achieved using a parallel architecture introduced as SONNET-MAP.

9.8 LTM representation of melodies in SONNET

Thus far, this chapter has shown how melodies can be represented at the input fields of SONNET modules. In contrast, this section illustrates SONNET's LTM phrase structure for learned melodies. As Section 5.3.3 explained, the categories formed at each cell assembly are encoded through the excitatory weights (z^{+}_{kjni}), which are considered as SONNET's LTM. Figure 9.13 illustrates the LTM structure of the first four segmented phrases of the melody *Across the Universe*. Each phrase contains four IOIs which are encoded by five excitatory weights. Note that although four phrases exist, only two cell assemblies are required to encode them. This is because three out of the four phrases contain identical IOI sequences. Furthermore, from a hierarchical point of view, only a single cell assembly at $\mathcal{F}_3$ is required to encode the entire phrase sequence. Thus, this way of organizing melodies in LTM is very efficient.

9.9 Summary

This chapter showed how it is possible to represent the characteristic features of a melody at $\mathcal{F}_1$ using a variety of schemes and techniques. IOIs, IPCs, pitch-intervals and melodic contours can be derived from an event field, $\mathcal{F}_0$, and represented at $\mathcal{F}_1$ using either the GPC scheme or the CAR scheme. Moreover, the manner in which melodies are represented in LTM was illustrated. Finally, we also briefly discussed the desirability of enabling more than one of these features to interact in order to process and recognize melodies in a more effective manner. The next chapter expands on this point and introduces a neural network architecture, known as SONNET-MAP, which addresses this issue.

Chapter 10

SONNET-MAP: Associating Parallel SONNET Modules

10.1 Introduction

The use of a single SONNET module (or hierarchy) limits its applicability to processing one-dimensional input patterns (see Section 9.7.2). Therefore, in order to process multidimensional input patterns, a new architecture is required. To achieve this, a number of issues need to be addressed, which relate specifically to the use of parallel architectures that process unsegmented temporal patterns.

This chapter introduces a network, known as SONNET-MAP, that addresses these issues. SONNET-MAP can automatically process unsegmented, multidimensional temporal patterns to form a parallel (and hierarchical) memory structure that encodes the significant patterns contained within an input stimulus (in this case, the pitch and rhythm dimensions of a melody). This is the first time that SONNET modules have been configured to operate in this way and will prove an invaluable precursor for work on melody segmentation and retrieval presented in Chapters 12 and 13.

10.2 Nomenclature

Before we describe the architecture and operation of SONNET-MAP, we introduce a number of new concepts and terms below:

The set $\mathcal{A}$: The set of input assemblies $\{\gamma_1, \gamma_2, \cdots, \gamma_{n-1}, \gamma_n\}$, which specify a pattern that is embedded within the input sequence at $\mathcal{F}_1$. $\mathcal{A}$ is used in *boundary point alignment* and *classification synchronization.* Its derivation is facilitated by the use of associative map fields that temporarily connect input assemblies from different $\mathcal{F}_1$ fields. Associations only form between input assemblies at different dimensions if these input assemblies fired in response to different dimensions of the same note onset (see Section 10.5.2).

The set $\mathcal{M}$: The set of cell assemblies $\{\chi_1, \chi_2, \cdots, \chi_{n-1}, \chi_n\}$, which match the pattern specified by the set $\mathcal{A}$. A cell assembly χ_i is considered to match $\mathcal{A}$ if its set T_i is *aligned* with $\mathcal{A}$ and its LTM weight vector matches its STM weight vector ($M_i \geq K_M$).

Alignment: A cell assembly χ_i is considered to be aligned with $\mathcal{A}$ if $T_i = \mathcal{A}$.

Boundary point alignment: A process which guarantees that every dimension of a phrase is learned. Boundary point alignment is necessary when parallel hierarchies are used to process unsegmented input patterns.

Classification synchronization: A process which ensures that every dimension of a learned phrase is chunked-out at approximately the same time. Classification synchronization is necessary when parallel and hierarchical architectures are used.

10.3 SONNET-MAP overview

SONNET-MAP (see Figure 10.1) is composed of two parallel SONNET modules; one which represents pitch sequences and another which represents rhythms. Since multiple SONNET modules are used within a single framework, an additional alphabetic index is used to distinguish the fields, cells and weights of separate modules where ambiguities may arise. In this thesis $\mathcal{F}_1^a$ and $\mathcal{F}_2^a$ represent the input field and the masking field of the pitch module, while $\mathcal{F}_1^b$ and $\mathcal{F}_2^b$ represent the input field and the masking field of the rhythm module. Furthermore, all input assemblies and cell assemblies at $\mathcal{F}_1^a$, $\mathcal{F}_2^a$, $\mathcal{F}_1^b$ and $\mathcal{F}_2^b$ are indexed as follows:

$\mathcal{F}_1^a$ input assemblies are indexed using m and j, where γ_{mj} represents the m^{th} input assembly within the j^{th} sibling group.

$\mathcal{F}_2^a$ cell assemblies are indexed using n and i, where χ_{ni} represents the n^{th} cell assembly within the i^{th} sibling group.

$\mathcal{F}_1^b$ input assemblies are indexed using k and l, where γ_{kl} represents the k^{th} input assembly within the l^{th} sibling group.

$\mathcal{F}_2^b$ cell assemblies are indexed using p and q, where χ_{pq} represents the p^{th} cell assembly within the q^{th} sibling group.

These modules are connected via associative maps, derived from the ARTMAP architecture [21], that associate these modules at the note level and at the phrase level. At the note level, temporary associations are formed between input assemblies at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ that activate in response to different features of the same note onset. Meanwhile, at the phrase level, pitch sequence categories are bound to their corresponding rhythm categories. These associations allow the construction of rigid, two-dimensional representations of melodic phrases. In addition, a further SONNET module may be utilized to aggregate the phrases learned by the parallel modules. This forms a parallel and hierarchical memory structure which encompasses the significant patterns contained within a melody. With such an organization, a single classification cell at the most abstract level in the hierarchy represents the entire melody. In order for this configuration to work properly, a number of technical and practical issues need to be overcome. Most of these issues are interdependent and, therefore, must be considered with respect to one another.

Consistency of working memory. Since multiple input fields are used to represent input sequences, with each field representing a distinct input dimension, the entire collection of input fields constitute SONNET-MAP's working memory. In this thesis, a melody is represented by the dimensions of pitch and rhythm, therefore $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ represents SONNET-MAP's entire working memory. Since two input assemblies activate in response to each note onset, mechanisms are required to ensure the coordinated reset of these input assemblies. Otherwise, due to input field overloads and the chunking-out process, the STM memory trace for the melody will become inconsistent across the entire working memory. This issue is discussed at greater length in Section 10.4.

Forming consistent boundary points. Unless a set of well-defined interactions are specified between the input fields ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) and the classification fields ($\mathcal{F}_2^a$ and $\mathcal{F}_2^b$) of each parallel SONNET module, the categories formed at each of these fields may conflict. Since the input sequences that are presented to SONNET-MAP are unsegmented, it is highly unlikely that every-single dimensional phrase formed at *module a* will be aligned along the same boundary points as *module b* (and vice versa). This is because different grouping cues at $\mathcal{F}_1^a$ (pitch-based) and $\mathcal{F}_1^b$ (rhythm-based) determine the boundary points at those modules. Unless this issue is addressed, the overlapping nature of the learned categories at each module will hinder the formation of multidimensional phrase representations. This prevents the development of a meaningful phrase hierarchy and the maintenance of a consistent working memory. Issues relating to boundary point alignment are discussed further in Section 10.5.

Forming multidimensional phrase representations. A solution to the boundary point problem does not imply the synchronous chunking-out of the categories formed at these boundary points. This is because the parallel modules would be still operating in isolation from one another in terms of actually performing the chunking-out of a pattern. As a consequence, contradictory chunk-outs will lead to an inconsistent working memory and its related problems. Furthermore, unless multidimensional phrase representations can form, neither multidimensional phrase hierarchies nor associative priming can be utilized. Therefore, a method of binding categories that represent different dimensions of the same melodic phrase is required. These issues are discussed further in Section 10.6.

Forming a hierarchical phrase structure. Once the problems discussed so far have been addressed, a multidimensional phrase hierarchy can form. However, unlike the SONNET hierarchy described in Chapter 7, input to a single $\mathcal{F}_3$ classification field will originate from two different sources; a pitch sequence classification and a rhythm classification. Although

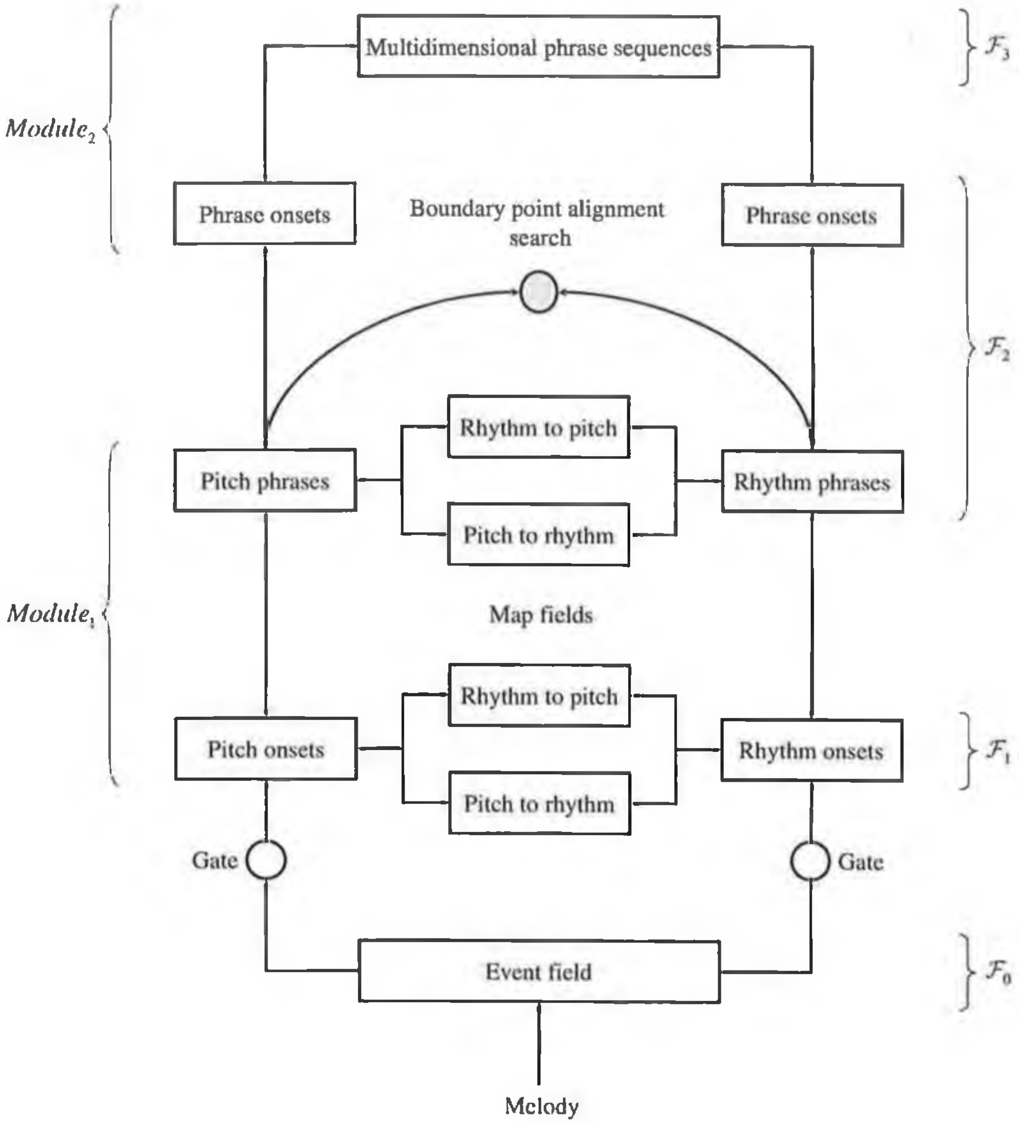


Figure 10.1: The SONNET-MAP architecture with a cascaded SONNET module.

only minor modifications are required, this is the first time that such a structure has been introduced (see Section 10.7).

Optimization of resources. To allow SONNET-MAP to process larger input sequences, the optimization of resources becomes an important requirement. Section 10.8 shows how SONNET-MAP can be empowered to dynamically determine its own size depending on how learning progresses.

As far as we are aware, the work presented in this chapter represents the first self-organizing neural network capable of autonomously learning unsegmented, temporal, multidimensional input patterns using parallel SONNET modules. Furthermore, Chapters 11 and 13 will show how this approach is beneficial for CBR. However, for now, the remainder of this chapter will focus on the details of how each of the issues raised in this section are addressed.

10.4 Synchronizing γ resets across $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$

Due to the parallel configuration of SONNET modules within the SONNET-MAP architecture, *separate* pitch and rhythm input fields (labelled $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ respectively) will respond to the same melody, albeit to its different dimensions. Therefore, each note onset in a melody will activate two input assemblies; one in each input field. In this respect, both input fields, $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$, are to be considered as SONNET-MAP's working memory, which forms a distributed representation of each melody by virtue of the fact that two input assemblies represent each note onset. However, difficulties may arise if input assembly resets are not synchronized across $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$, such that the *input assemblies that have activated at the same time in response to the same note onset are also reset at the same time.* Remember, an input assembly may be reset by either an input field overload (see Section 5.3.1) or a cell assembly chunk-out (see Section 5.3.8). The coordination of input field resets helps maintain a consistent representation of the input stimulus in working memory, which is fundamental to addressing the issues presented in Section 10.3.

10.4.1 Inconsistencies due to input field overload mechanisms

In order to illustrate this problem, consider the situation that occurs when the following overload schemes are operating together:

1. Suppose the input field representing pitch sequences (specifically, pitch-intervals at $\mathcal{F}_1^a$) has its input parameters set such that $\mathcal{F}_1^a$ overloads when seven note onsets have been presented. Furthermore suppose that, when this occurs, three or four (with 50% probability) of the oldest active input assemblies are reset.

2. On the other hand, suppose that the rhythm module (specifically, IOI's at $\mathcal{F}_1^b$) is configured in the manner described by Roberts [133, Chapter 6]. That is, each input assembly is reset after it has been active for six tactus-spans.

Furthermore, suppose that the notes of the melody shown in Fragment 9.1 are presented to SONNET-MAP, giving the activity traces shown in Figure 10.2. After the final IOI has been completed, the oldest event onset at $\mathcal{F}_1^b$ will reach its overload threshold, causing the deletion of this event onset from $\mathcal{F}_1^b$. Since $\mathcal{F}_1^a$ will not have reached its overload threshold yet, no resets will have occurred there. This situation gives rise to the activity trace shown in Figure 10.3. At this point, working memory is inconsistent, which hinders the correct operation of the boundary point alignment process (see Section 10.5).

Both overload schemes are reasonable and appropriate when they operate in isolation from one another. However, when they are integrated within an interacting parallel architecture, these overload schemes conflict and lead to inconsistencies between the input fields representing the melody. To address this problem, we found it was beneficial to initially allow the SONNET-MAP input fields ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) to operate independently, but only for a specified number of epochs. This allows pitch sequences and IOI sequences to form independently from one another, which enables the nuances of each representation to contribute to the formation of patterns.

Then, after the specified number of epochs when patterns have had a chance to form at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$, both overload reset mechanisms are disabled. At this point, input cells at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ will be reset solely on the basis of the chunking-out processes at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$. Additionally, to ensure consistency at this point, any input cells that are older than the oldest input cell just chunked-out are also reset. Further consistency is ensured by the boundary point alignment and classification synchronization processes specified in Sections 10.5 and 10.6 respectively.

10.4.2 Inconsistencies due to chunking-out

Another source of inconsistencies across SONNET-MAP's working memory is chunking-out. Suppose a cell assembly at $\mathcal{F}_2^a$ recognizes the occurrence of its pattern across $\mathcal{F}_1^a$, and as a consequence, classifies this pattern. This will result in the reset of input assemblies at $\mathcal{F}_1^a$ ($\gamma_{mj}^a \in T_{ni}^a$) due to the chunking-out process. If the rhythm at $\mathcal{F}_2^b$, that corresponds to this pitch sequence at $\mathcal{F}_2^a$, has not yet been learned, or has not yet chunked-out its pattern at $\mathcal{F}_1^b$, inconsistencies across SONNET-MAP's working memory will occur (see Figure 10.4).

To ensure that chunking-out does not cause inconsistencies across SONNET-MAP's working memory, two conditions must be satisfied before a cell assembly chunks-out its pattern:

1. All the other dimensions of the pattern must be learned. That is, if a pitch sequence is learned at $\mathcal{F}_2^a$, its corresponding rhythm at $\mathcal{F}_2^b$ must also be learned (and vice versa).

2. All the other dimensions of the pattern must also be chunked-out. That is, if a pitch sequence is chunked-out at $\mathcal{F}_2^a$, its corresponding rhythm at $\mathcal{F}_2^b$ must also be chunked-out (and vice versa).

These problems are not easily solved and will be discussed in Sections 10.5 and 10.6 respectively.

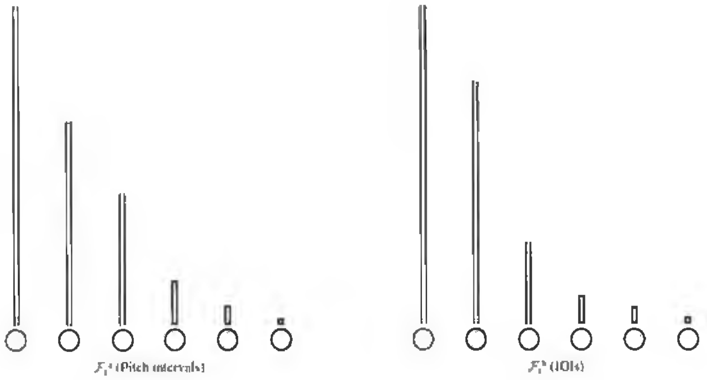


Figure 10.2: The state of SONNET-MAPs working memory ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) in response to the presentation of Fragment 9.1.

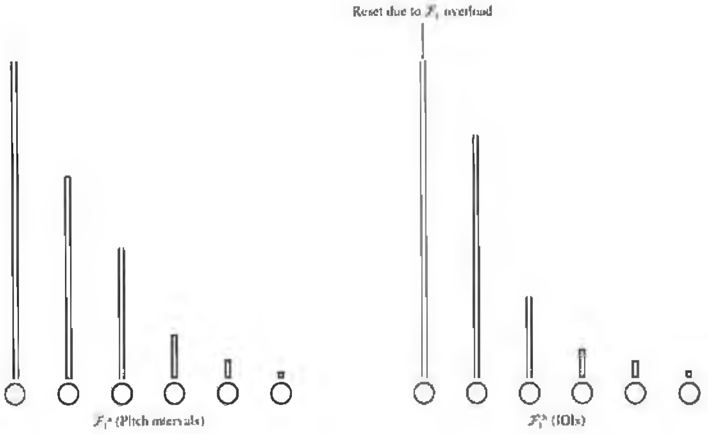


Figure 10.3: Conflicting overload schemes at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ lead to an inconsistent melody representation across SONNET-MAP's working memory. In this case, the oldest event onset at $\mathcal{F}_1^b$ has reached its overload threshold and has therefore been reset.

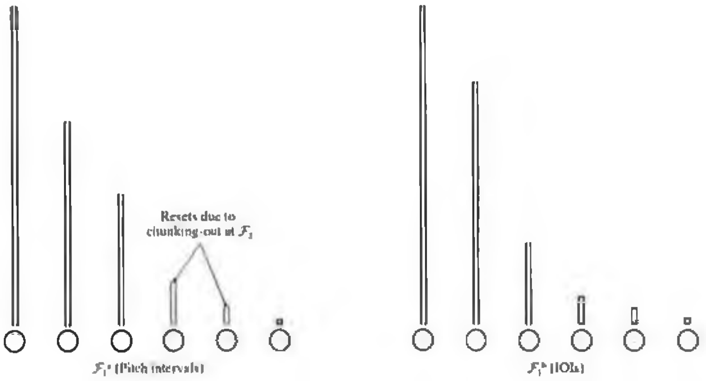

Resets due to chunking-out at $\mathcal{F}_2$
$\mathcal{F}_1^a$ (Pitch intervals)
$\mathcal{F}_1^b$ (IOIs)

Figure 10.4: Unless both the pitch and rhythm dimensions of a melodic phrase are chunked-out at approximately the same time, an inconsistent working memory will result. In this case, the pattern consisting of the 4^{th}, 5^{th} and 6^{th} notes are recognized causing the 4^{th} and 5^{th} note onsets to be deleted from $\mathcal{F}_1^a$. Remember, using interval encoding schemes, the last onset of a pattern is not deleted.

10.5 Boundary point alignment

In Section 10.3 the requirement of forming consistent boundary points across each dimension of an input stimulus was illustrated. It is imperative that this issue be addressed; otherwise the problems illustrated in Section 10.3 will arise. For example, inconsistent boundary points will lead to an inconsistent working memory. Conflicting classifications will hinder the formation of multidimensional phrase categories, which in turn prevents the formation of meaningful phrase hierarchies. Furthermore, SONNET-MAP would be unable to process unsegmented input patterns. This would negate the entire justification for using SONNET modules in the first place.

This section introduces a mechanism called *boundary point alignment* to address these problems. Boundary point alignment is only allowed to occur during a single epoch and is initiated when a cell assembly in any dimension chunks-out its pattern during this epoch. Boundary point alignment ensures that every dimension of a melodic phrase is learned when separate modules are used to process different dimensions of the phrase. Consequently, only cell assemblies which contribute to the segmentation of a pattern will have their associated patterns at other dimensions learned. It is essential when a pattern is learned and subsequently chunked-out in one dimension (for example, a pitch sequence at $\mathcal{F}_2^a$) that its alternate dimension also be learnt and chunked-out *immediately* (that is, the pitch sequence's rhythm, at $\mathcal{F}_2^b$). As a result, phrase categories that have formed at different, but parallel, SONNET modules are aligned along the same boundary points. Furthermore, boundary point alignment ensures that all of the dimensions of a melodic phrase are learnt, regardless of which classification field first learns its own dimension of the phrase. This immediacy removes the potential for any future inconsistencies. In particular, this process facilitates the binding of different dimensions of a single phrase through the process of classification synchronization, which is discussed in Section 10.6.

The chunking-out of a cell assembly's pattern is monitored by an *orienting subsystem*, which ensures that every dimension of the pattern is learned. On chunking-out, a boundary point alignment signal is triggered by the orienting subsystem. This signal initiates a search process (analogous to the ART search process described in [15]) in each of the other dimensions. Provided the pattern has not been learned already at these dimensions, the search process will find cell assemblies that are suitable for learning the pattern. The pattern to be learned is specified by the set $\mathcal{A}$, which is specific to each dimension; $\mathcal{A}^a$ and $\mathcal{A}^b$. The set $\mathcal{A}$, although analogous to the set T, is a module-wide set and is used by every cell assembly to determine its suitability for learning. When the search process establishes the choice of cell assembly to learn the pattern specified by $\mathcal{A}$, a fast-learn process is initiated that encodes the pattern there and then. The set $\mathcal{A}$ is constructed using associative map fields at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$, the details of which are described next. See Figure 10.5 for an overview of the boundary point alignment process.

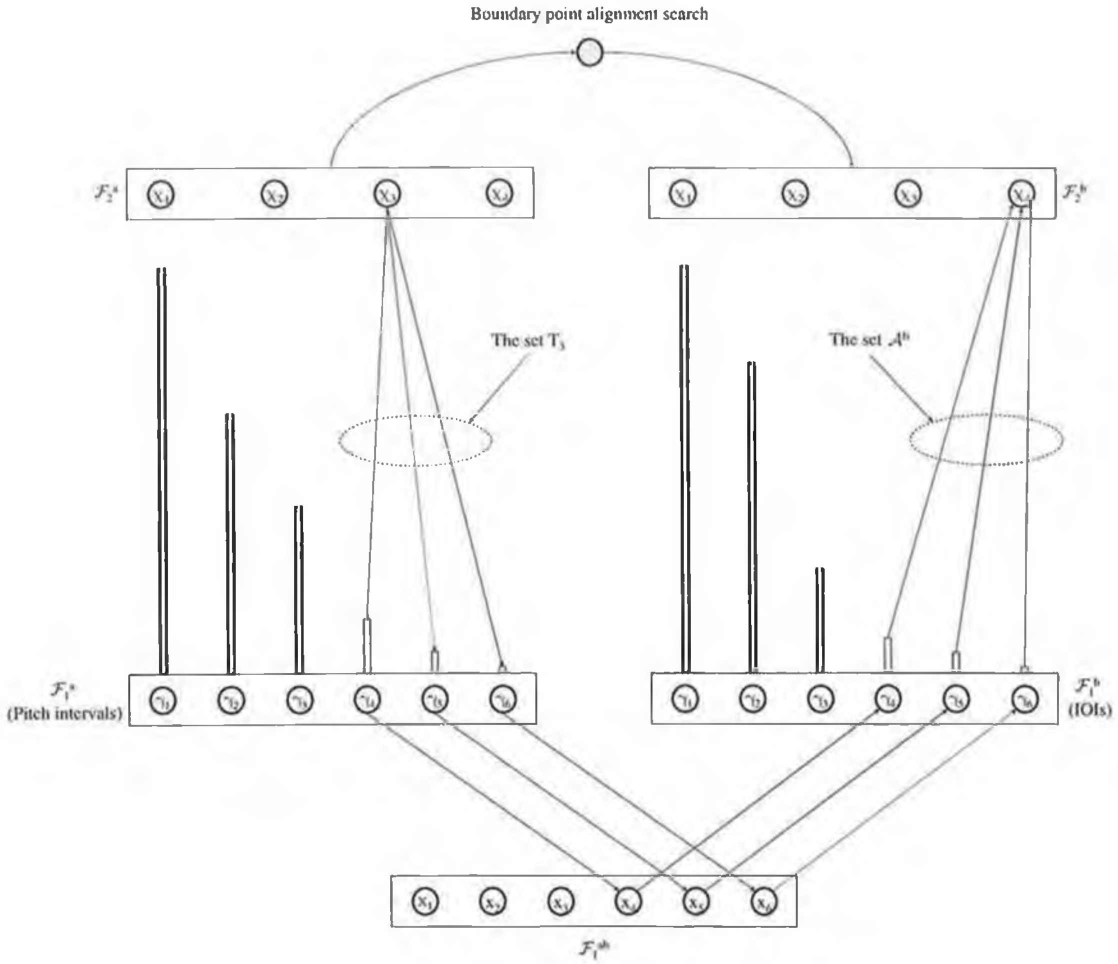


Figure 10.5: The boundary point alignment process. The note onsets represented at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ are from Fragment 9.1. The chunking-out process is monitored by an orienting subsystem, which ensures that every dimension of a melodic phrase is learned and chunked-out. In this example, the pitch-interval sequence represented by the 4^{th}, 5^{th} and 6^{th} note onsets have been learned at $\mathcal{F}_2^a$ and are in the process of being chunked-out. This causes the pitch-interval sequence's corresponding rhythm to be learned and subsequently chunked-out. This ensures that consistent boundary points are formed across SONNET-MAP's entire working memory.

10.5.1 Temporarily associating input assemblies

In order for boundary point alignment to work, it must be possible to identify the specific input assemblies at $\mathcal{F}_1^b$ that fired in response to the same note onsets comprising any pattern learnt by $\mathcal{F}_2^a$, and vice versa. These sets of cells are labelled $\mathcal{A}^a$ and $\mathcal{A}^b$ respectively. Note that $T_{ni}^a = \mathcal{A}^a$ if the pattern was classified at $\mathcal{F}_2^a$. In effect, for a two-dimensional hierarchy, only the construction of the set $\mathcal{A}^b$ is required in response to a category forming at $\mathcal{F}_2^a$ (although, the set $\mathcal{A}^a$ will be created by default due to the signals that arrive from $\mathcal{F}_2^a$). Furthermore, it is important to point out that this step would not be necessary if we were dealing with pre-segmented input patterns because each melodic phrase would be presented in isolation, rather than embedded within larger patterns. However, since we are dealing with an unsegmented stream of input events, the pattern of interest will generally be embedded within this stream; thus the problem is more challenging.

Two associative maps are employed between each input field in order to construct the sets $\mathcal{A}^a$ and $\mathcal{A}^b$. These associative maps require the following properties:

Temporary associations. Input assemblies that respond to the same note onset need to be temporarily associated. Associations are temporary because input assemblies are only active for a short period of time in working memory. Consequently, when an input assembly is reset in one dimension, it is unlikely, upon re-activation, to be associated with the same input assembly at the other dimension. Therefore, learned associations must be *unlearned* at the same time as the input assemblies at either end of the association are reset. This can be regarded as a form of active forgetting, which is considered by many researchers, as a necessity for complex behaviour to occur.

Fast learning. Unless the formation of temporary associations occurs quickly (in fact, during the time period of K_{latch}), it may be impossible at a later time to form these associations. This is due to the fact that numerous input assemblies at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ will be active simultaneously. In fact, temporary associations are formed at the instant an event onset causes a set of input assemblies across $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ to activate.

One-to-one correspondence. Each active input assembly at $\mathcal{F}_1^a$ should only be associated with a single active input assembly at $\mathcal{F}_1^b$ and vice versa. Consequently, a one-to-one correspondence is said to exist between such input assemblies. This differs from the associative map field that will be described in Section 10.6, where each category formed may be associated with multiple categories at another dimension. For example, a single rhythm may be associated with many pitch sequences.

Binary operation. Binary latch cells, at each input assembly, are responsible for communicating the occurrence of events at that input assembly to its associated map fields (whether it is a note event or a chunk-out event). The operation of the associative map field is also binary, because associative cells are either active or inactive. Similarly, since the associative weights converge to the map field's binary activity pattern, associative weights are also binary. This binary operation facilitates the fast creation of temporary associations.

Priming. In order to construct the set $\mathcal{A}$ at each dimension, associative priming signals need to be propagated via the learned associations between input fields. These priming signals are instigated by the chunking-out signals sent via the feedback links at the time a cell assembly classifies its pattern.

The map fields described here are derived from those introduced in Carpenter et al. [21] and described in Chapter 4. Consequently, where appropriate, the terminology will remain consistent. Furthermore, the input fields $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ are connected via two associative maps — one in each direction, $\mathcal{F}_1^a \rightarrow \mathcal{F}_1^b$ ($\mathcal{F}_1^{ab}$) and $\mathcal{F}_1^b \rightarrow \mathcal{F}_1^a$ ($\mathcal{F}_1^{ba}$). However, since their architecture and operation are identical, their behaviour will be described in terms of the map field from $\mathcal{F}_1^a \rightarrow \mathcal{F}_1^b$ (see Figure 10.6).

Map field architecture

For the map field architecture to form temporary associations between input assemblies, two additional cells at each input assembly are introduced (see Figure 10.7). The first is a latch cell, labelled l, which behaves in the same way as the intermediate latch cells used in cascading SONNET architectures (see Section 6.4). That is, these latch cells are activated (and set to unity) by the occurrence of a note onset at $\mathcal{F}_0$, and remain active for a brief time period of K_{latch}. Input cells s are now activated by the latch cells rather than directly by $\mathcal{F}_0$. Apart from this minor change, there is no difference in the way input cells operate. Note that K_{latch} must be sufficiently small so that multiple latch cells will not be simultaneously active at any one input field due to input sequences that are presented rapidly.

Additionally, a prime cell a is also utilized. Prime cells behave in a similar manner to latch cells. Once activated (to unity) they remain active for a brief time period, which is also specified by K_{latch}. Prime cells may be activated in either of two ways; via l due to a note onset at $\mathcal{F}_0$ or via d caused by a chunk-out at $\mathcal{F}_2$. Once activated, prime cells will either cause associative learning or associative priming, depending on the state of the network at that time. The use of a enables priming to occur without affecting the activation levels of the input cells in any way.

$\mathcal{F}_1^{ab}$, which associates $\mathcal{F}_1^a$ with $\mathcal{F}_1^b$, consists of $N_1^{g,b}$ sibling groups each containing $N_1^{s,b}$ siblings (or associative cells), labelled x_{kl}. There is a one-to-one correspondence between input assemblies at $\mathcal{F}_1^b$ and each map field's associative cell (see Figure 10.6). A fixed connection linking a_{kl}^b to x_{kl} communicates the occurrence of an event at $\mathcal{F}_1^b$. A reverse connection linking x_{kl} to a_{kl}^b is used to send associative priming signals, in order to construct the set $\mathcal{A}^b$. Each a_{mj}^a cell at $\mathcal{F}_1^a$ is connected to every $\mathcal{F}_1^{ab}$ cell by adjustable excitatory weights, labelled u_{mjkl}, which communicate event occurrences at $\mathcal{F}_1^a$ to $\mathcal{F}_1^{ab}$. Remember, only the map field, $\mathcal{F}_1^{ab}$, is being described here.

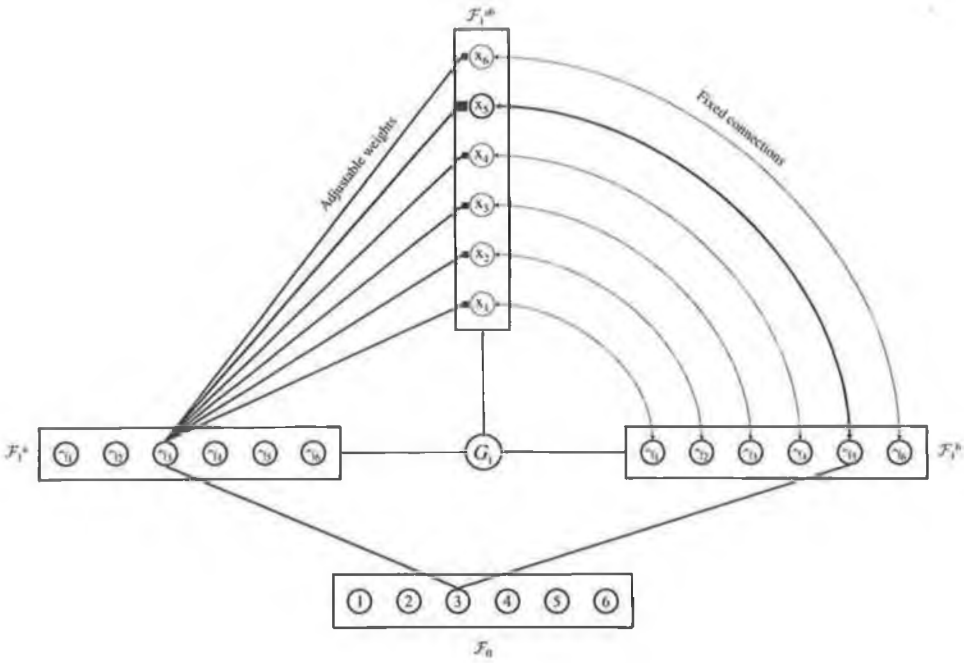


Figure 10.6: The associative map between $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ is used to form temporary associations between input assemblies that respond to the same note onset. In this example, the activation of the the event field due to a note occurrence causes γ_3^a at $\mathcal{F}_1^a$ to activate and γ_5^b at $\mathcal{F}_1^b$ to activate. This in turn causes a temporary association to form between γ_3^a and γ_5^b via the associative map field $\mathcal{F}_1^{ab}$.

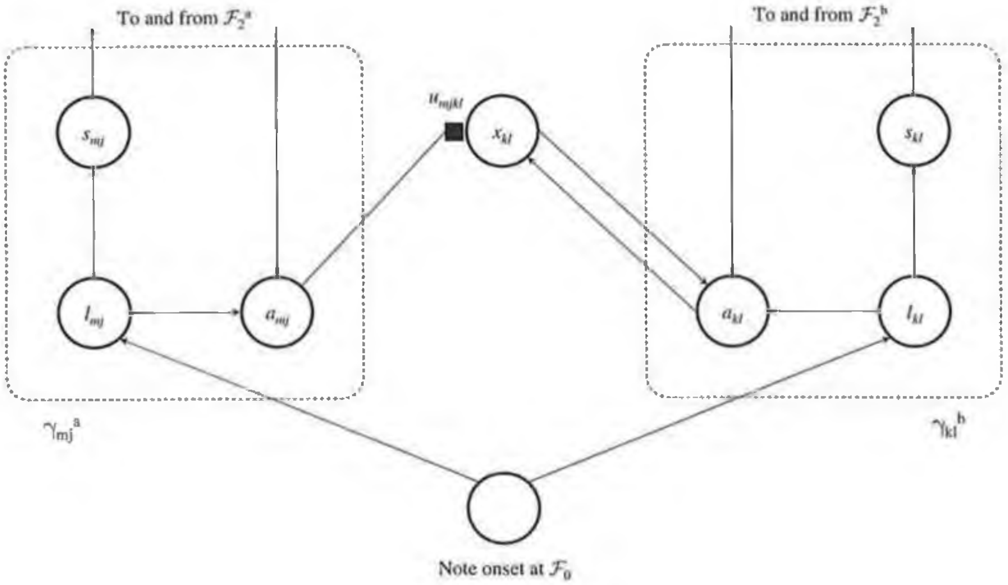


Figure 10.7: Two additional cells (l and a) are used at each input assembly to facilitate the temporary associations between input assemblies that respond to the same event onset.

Map fields are event-driven

Since these map fields respond directly to event occurrences, they are considered as event-driven (see Section 6.3.2). Events that cause map field cells to activate take two distinct forms:

Note events at $\mathcal{F}_0$ cause associative learning. A latch cell activates in response to a note onset at $\mathcal{F}_0$. This in turn causes its corresponding prime cell to activate. When an input field responds to an event in this way, its corresponding map fields are activated for a brief time period. Since an event onset results in two input assemblies activating (one at $\mathcal{F}_1^a$ and one at $\mathcal{F}_1^b$), an association between these input assemblies will form.

Chunk-out signals from $\mathcal{F}_2$ cause associative priming. The chunking-out of a cell assembly χ_{chunk} at $\mathcal{F}_2^a$ will cause the prime cells at $\mathcal{F}_1^a$, which are members of T_{chunk}, to activate in response. Prime cells combine with learned associations at $\mathcal{F}_1^{ab}$ in order to attentionally prime any associated input assemblies at $\mathcal{F}_1^b$. The primed input assemblies at $\mathcal{F}_1^b$ become members of the set $\mathcal{A}^b$.

Associative learning and priming are considered to take place so rapidly that no other network component will evolve during this minute time span. Consequently, K_{latch} must be sufficiently small to accommodate this. The idea of differing time scales is not new to neural network research. In fact, the search process in the ART 1 neural network is considered to take place on a much quicker time scale than learning [21]. Since associative learning and priming take place on such a small time scale, the chunking-out of other cell assemblies will not interfere with these processes.

Associative cell activations

The activation of each $\mathcal{F}_1^{ab}$ associative cell is governed by a 2/3's rule, similar to that employed by ART 1 neural networks. That is, each associative cell can receive input from three different sources: $\mathcal{F}_1^a$, $\mathcal{F}_1^b$ and a binary gain control signal G_1. For an associative cell to activate, two out of three of these input signals must be active. Furthermore, associative cells may only evolve during the brief time period of K_{latch} following an event's occurrence. This is due to the event-driven operation of the map field. The activation equation for each $\mathcal{F}_1^{ab}$ associative cell, x_{kl}, is given by:

$$x_{kl} = \begin{cases} 1 & \text{if } a_{kl}^b + I_{kl}^{+^{ab}} + G_1 \geq 2 \\ 0 & \textit{otherwise} \end{cases} \tag{10.1}$$

$$I_{kl}^{+^{ab}} = \sum_{j=1}^{N_1^{g,a}} \sum_{m=1}^{N_1^{s,a}} a_{mj}^a u_{mjkl} \tag{10.2}$$

$$G_1 = \begin{cases} 0 & \text{if } \mathcal{F}_1^a \text{ and } \mathcal{F}_1^b \text{ are both active} \\ 1 & \textit{otherwise} \end{cases} \tag{10.3}$$

where a_{kl}^b is the input on the fixed link from $\mathcal{F}_1^b$. $I_{kl}^{+^{ab}}$ is the gated input received via the adjustable weights from each prime cell, a_{mj}^a, at $\mathcal{F}_1^a$. Finally, G_1 is a binary gain control signal, which is based on the activity of both $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$. In the scenario presented here, $\mathcal{F}_1^a$ or $\mathcal{F}_1^b$ are considered as being active if any one of their priming cells are active. Since the only points in time of interest are when note events or chunking-out events occur, the input cells are not considered because they remain active for relatively long periods. Consequently, the brief activation of prime cells are used to trigger the formation of temporary associations.

Map field learning

The temporal overlap resulting from the simultaneous activation of input assemblies at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ (for example, γ^a and γ^b), will cause an $\mathcal{F}_1^{ab}$ associative cell to activate. This in turn causes the formation of a temporary association between these two input assemblies. This is achieved by adjusting the connection weights from $\mathcal{F}_1^a$ to $\mathcal{F}_1^{ab}$. Initially, each of these adjustable weights, u_{mjkl}, is set to unity. Thereafter it is modified according to the following outstar learning rule:

$$\frac{\delta}{\delta t} u_{mjkl} = a_{mj}^a u_{mjkl}(x_{kl} - u_{mjkl}) \tag{10.4}$$

In effect, the adjustable weight vector $\mathbf{u}_{mjkl}$ (from $\mathcal{F}_1^a$ to $\mathcal{F}_1^{ab}$) tracks the $\mathcal{F}_1^{ab}$ activity vector $\mathbf{x}^{ab}$, provided an $\mathcal{F}_1^a$ prime cell is active.

Note, however, before any learning has occurred, the activation of a priming cell at $\mathcal{F}_1^a$ in the absence of an active priming cell at $\mathcal{F}_1^b$ would cause every $\mathcal{F}_1^{ab}$ cell to activate. This is due to the initialization of $\mathbf{u}^{ab}$ to unity. This, in turn, would cause every $\mathcal{F}_1^b$ prime cell to activate, which is clearly undesirable. However, this situation never occurs because an associative priming cell only ever becomes active in following circumstances:

In response to an event onset at $\mathcal{F}_0$. In this circumstance, a single priming cell at both $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ will become active. This in turn causes a single associative cell at $\mathcal{F}_1^{ab}$ to become active (see Equation 10.1). Consequently, the weight vector $\mathbf{u}^{ab}$ will converge to this activity pattern at $\mathcal{F}_1^{ab}$, thus forming a temporary association (see Equation 10.4).

In response to a cell assembly chunking-out at $\mathcal{F}_2^a$. In this case, multiple priming cells at $\mathcal{F}_1^a$ will activate. However, temporary associations will have already been formed due to the previous activation of prime cells when a note onset occurred. Remember, this note onset is now contributing to the chunk-out at $\mathcal{F}_2^a$. This activation of priming cells at $\mathcal{F}_1^a$ will activate multiple associative cells at $\mathcal{F}_1^{ab}$, which will lead to associative priming at $\mathcal{F}_2^b$ and the construction of the set $\mathcal{A}^b$.

As outlined, a major property of the map field at this level is that associations are only temporary. In fact, associations should only last as long as the input cells themselves remain active. Therefore, when an input assembly γ_{mj} is reset, the corresponding adjustable connections to $\mathcal{F}_1^{ab}$ should also be reset. As opposed to being reset to zero, as in the case of s_{mj}, the adjustable weights u_{mjkl} are reset to their original values of unity. This allows them to be used again to form other associations between input assemblies in response to other note onsets.

In summary, the adaptable weights from $\mathcal{F}_1^a \rightarrow \mathcal{F}_1^{ab}$ are modified such that the activation of an associative priming cell at γ_{mj}^a, in the absence of any input event at $\mathcal{F}_0$, will cause its associated input assembly γ^b_{kl} to be associatively primed. As Section 10.5.2 will show, input assemblies that are primed in this way will form the set $\mathcal{A}^b$.

Activation scenarios

Due to the map field dynamics, the following activation scenarios and subsequent responses are possible:

$\mathcal{F}^a_1$ active - $\mathcal{F}^b_1$ active. This situation arises when a note onset occurs at $\mathcal{F}_0$, thus activating the appropriate input assemblies at $\mathcal{F}^a_1$ and $\mathcal{F}^b_1$. Since both input fields are active, the binary gain control signal will be inactive ($G_1 = 0$) and a single associative cell at $\mathcal{F}^{ab}_1$ will activate (see Equation 10.1). In this circumstance, associative learning will form a link between the active input assembly at $\mathcal{F}^a_1$ and the active input assembly at $\mathcal{F}^b_1$ (see Equation 10.4).

$\mathcal{F}^a_1$ active - $\mathcal{F}^b_1$ inactive. This situation arises when a chunk-out event occurs at $\mathcal{F}^a_2$, which causes $a^a_{mj} \in T^a_{ni}$ to activate. Since only the $\mathcal{F}^a_1$ field becomes active, the binary gain control signal will activate to unity ($G_1 = 1$). In this circumstance, due to previous associative learning, multiple $\mathcal{F}^{ab}_1$ associative cells will activate (see Equation 10.1). These activations will prime $\mathcal{F}^b_1$ when constructing the set $\mathcal{A}^b$ (see Section 10.5.2).

$\mathcal{F}^a_1$ inactive - $\mathcal{F}^b_1$ active. This situation arises when a chunk-out event occurs at $\mathcal{F}^b_2$. Since only the $\mathcal{F}^b_1$ field becomes active, the binary gain control signal will activate to unity ($G_1 = 1$). In this circumstance, multiple associative cells at $\mathcal{F}^{ab}_1$ will activate due to the one-to-one fixed links from $\mathcal{F}^b_1$.

$\mathcal{F}^a_1$ inactive - $\mathcal{F}^b_1$ inactive. In this case, since neither input field is active, the binary gain control signal will be active ($G_1 = 1$). However, no associative cells at $\mathcal{F}^{ab}_1$ will activate.

10.5.2 Constructing the set $\mathcal{A}$ at each dimension

This section describes how chunking-out signals received at $\mathcal{F}^a_1$ from $\mathcal{F}^a_2$ can be used to associatively prime input assemblies at $\mathcal{F}^b_1$ via $\mathcal{F}^{ab}_1$. It is these associatively primed input assemblies at $\mathcal{F}^b_1$ that form the set $\mathcal{A}^b$. Priming operates in a similar fashion to ARTMAP priming. Since the problem to be solved is that of boundary point alignment, priming occurs at the instant a cell assembly at $\mathcal{F}^a_2$ begins chunking-out its pattern. Signals are fed back, via the feedback connections, to the associative priming cells at $\mathcal{F}^a_1$ that are members of the set T^a_{ni}. Each associative priming cell activates in response to this signal.

Due to the $\mathcal{F}^{ab}_1$ associative cell activation equation and prior learning (see Equations 10.1 and 10.4 respectively), a set of cells at $\mathcal{F}^{ab}_1$ will activate. These signals are then propagated, via the one-to-one connections, from $\mathcal{F}^{ab}_1$ to $\mathcal{F}^b_1$. There, an input assembly can determine if it is associatively primed by examining the state of its priming cell and latch cell. In general, a cell assembly is considered to be a member of the set $\mathcal{A}$ at its dimension if its prime cell is active while its latch cell is inactive:

$$\gamma_{kl} \in \mathcal{A}^b \iff a_{kl} = 1 \text{ and } l_{kl} = 0 \tag{10.5}$$

10.5.3 Initiating a search procedure on 'chunking-out'

The chunking-out process is monitored by an orienting subsystem similar to the non-specific attentional pulse used to govern SONNET's mode of operation (see Section 6.3). The orienting subsystem initiates a search process at $\mathcal{F}_2^b$ in response to a cell assembly chunking-out at $\mathcal{F}_2^a$. The ultimate goal is to ensure that all of the dimensions of a multidimensional phrase are learned. The search process at $\mathcal{F}_2^b$ finds the cell assembly (χ_{choice}) that best matches the input pattern specified by the set $\mathcal{A}^b$. Providing the pattern has not already been learned by any other cell assembly, χ_{choice} will *fast-learn* the pattern and commit to it using single-shot learning. Consequently, a pitch sequence at $\mathcal{F}_1^a$ and its corresponding rhythm at $\mathcal{F}_2^b$ are learned and independently represented. Note that the ART 1 subsystem initiates a search process in a similar way. However, the ART 1 search process occurs within a single module, whereas with SONNET-MAP, the search process is initiated at one module and occurs at another.

Once initiated, the search procedure checks to see if the pattern has already been learnt. If not, partially encoded patterns are checked. A partially encoded pattern is defined as having $z_{pq}^{tot} \geq K_5$. If no partially encoded patterns are currently learning the pattern, a cell assembly with a poor pattern representation is chosen. Such an assembly is defined as having $z_{pq}^{tot} < K_5$. The following formulation achieves this behaviour:

$$\chi_{choice} = \begin{cases} \max_{p,q\in\mathcal{M}}\left(z_{pq}^{tot}\right) & \text{if } \mathcal{M} \neq \emptyset \\ \min_{p,q\notin\mathcal{M}}\left(z_{pq}^{tot}\right) & \text{otherwise} \end{cases} \tag{10.6}$$

where $\mathcal{M}$ is the *match set*, which consists of cell assemblies that have learned or are attempting to learn the pattern specified by the set $\mathcal{A}^b$. A cell assembly must satisfy the following criteria to become a member of $\mathcal{A}^b$:

$$\chi_{pq}^b \in \mathcal{M}^b \iff T_{pq}^b = \mathcal{A}^b \text{ and } M_{pq}^b \geq K_M \tag{10.7}$$

In effect, a cell assembly's receptive field must equal $\mathcal{A}^b$. Furthermore, the normalized $\mathcal{F}_1^b$ activity pattern specified by $\mathcal{A}^b$ must also match the normalized excitatory weights specified by T_i. The use of z_{pq}^{tot} ensures that the cell assembly that has performed the greatest amount of learning will be chosen. If $z_{pq}^{tot} = 1$, the cell assembly has already learned the pattern. This is the best case scenario because no further effort needs to be made. In fact, it is likely that this will occur frequently because different pitch sequences are likely to have the same rhythm (and vice versa). This is one of the advantages of using parallel hierarchies over the scheme proposed by Lewis [88]; a more compact melody representation is learned. Furthermore, if no cell assembly matches the desired criteria, the cell assembly with the least amount of learning is chosen so that as little interference as possible with any previous learning occurs.

10.5.4 Performing the alignment

In the case where no cell assembly has learned the pattern specified by the set $\mathcal{A}^b$, the boundary point alignment process will initiate a fast-learn pulse, which causes χ_{choice} to learn the pattern specified by $\mathcal{A}^b$ by modifying its LTM state. The following steps are required:

1. If $T_{choice} \neq \mathcal{A}^b$, then set $T_{choice} = \mathcal{A}^b$. This can be achieved by setting all of χ_{choice}'s secondary weights in the set $\mathcal{A}^b$ to unity, while setting every other secondary weight to zero. At this point, the normalized input vector S_{choice} should be re-computed.
2. The excitatory weight vector z^+_{choice} should converge to the normalized input vector. This can be achieved by setting $z^+_{choice} = S_{choice}$. At this point, $I^\times_{choice}$ should be re-computed.
3. D_{choice} should converge to $I^\times_{choice}$ (that is, $D_{choice} = I^\times_{choice}$).
4. Set $\chi^{com}_{choice} = true$ to indicate that learning is now complete at χ_{choice}.

When the alignment has been performed, χ_{choice} chunks-out its pattern. Chunking-out should occur whether or not χ_{choice} had previously learned its pattern or not. As a consequence of chunking-out χ_{choice}, and the cell assembly at $\mathcal{F}^a_2$ which initiated the search process, associations will form between $\mathcal{F}^a_2$ and $\mathcal{F}^b_2$. This is achieved using the classification synchronization process which will be discussed next.

10.6 Classification synchronization

The previous section showed how both the pitch and rhythm dimensions of a melodic phrase can be learned and chunked-out using the boundary point alignment process. In addition to contributing to the maintenance of a consistent working memory, this process ensures that each dimension of such a phrase is learned at the same time. This guarantees that any SONNET module at $\mathcal{F}_3$ will be able to learn and recognize such pitch-rhythm pairings; thus forming a multidimensional phrase hierarchy.

However, is not desirable to operate the boundary point alignment search process indefinitely, particularly once a stable segmentation of the input sequence has been formed. Consequently, there comes a time when it is more efficient to use learned associations between $\mathcal{F}^a_2$ and $\mathcal{F}^b_2$. The use of these learned associations should guarantee that every dimension of the phrase is chunked-out in a reliable manner, maintain the consistency of SONNET-MAP's working memory and facilitate the formation of multidimensional phrase hierarchies. In addition, due to the formation of associations between pitch sequences and their corresponding rhythms, multidimensional phrase representations are formed across $\mathcal{F}^a_2$ and $\mathcal{F}^b_2$. As Chapter 13 will show, this organization proves invaluable for the CBR of melodies. This section introduces the term *classification synchronization* to describe the process whereby associations are formed between $\mathcal{F}^a_2$ and $\mathcal{F}^b_2$ and how these associations may be utilized.

Classification synchronization ensures that every dimension of a multidimensional phrase is classified and chunked-out at the same time. Associative maps are used to *bind* the cell assemblies that represent the different dimension of a pattern together. In this case, pitch sequence representations at $\mathcal{F}^a_2$ are bound to their corresponding rhythm representations at $\mathcal{F}^b_2$. Then, the neural pathways that exist between each dimension may be used to perform associative priming. Priming signals can be used to ensure that classifications are synchronized across all dimensions. This is achieved using two associative map fields ($\mathcal{F}_2^{ab}$ and $\mathcal{F}_2^{ba}$). The manner by which these associative maps operate is described in the remainder of this section.

10.6.1 Binding cell assemblies using associative map fields

This section addresses the problem of how the independent representations of a melodic phrase (in this case, the phrase's pitch sequence and rhythm) can be bound, thus forming a unified representation of the phrase. This problem, known as the *binding problem*, can be addressed using two associative maps; associating pitch sequences with their corresponding rhythms and vice versa. These associative maps require the following properties:

Permanent associations. Unlike the requirement to form temporary associations between input assemblies at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$, associations between $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ should be permanent. This is because the stable category codes formed by $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ cell assemblies are persistent, unlike the activation of input assemblies, which respond to note onsets only briefly. Furthermore, any particular phrase will always consist of the same pitch sequence and rhythm.

Multiple associations. Again, unlike the one-to-one associations between input assemblies at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$, a single cell assembly (or sibling group) at $\mathcal{F}_2^a$ may have associations with many cell assemblies (or sibling groups) at $\mathcal{F}_2^b$. This is because a single pitch sequence may be associated with many rhythms and vice versa. It is important to point out that, although a single cell assembly at $\mathcal{F}_2^a$ may be associated with many cell assemblies at $\mathcal{F}_2^b$, a single melodic phrase is uniquely defined by two cell assemblies; one at $\mathcal{F}_2^a$ and one at $\mathcal{F}_2^b$. Furthermore, the use of multiple associations also implies the use of multiple predictions when associative priming is applied. Due to this requirement, the operation of the map field is markedly different to that used for boundary point alignment.

Fast learning. Due to boundary point alignment, each classification field will learn its own dimension of a melodic phrase in response to chunking-out events. In response, a prime cell (similar to that used for boundary point alignment) will activate in response to each chunk-out. Since prime cells only activate for a brief time period, associations between these cell assemblies only have a short time period during which to learn. Therefore, learning must occur quickly.

Binary operation. Binary latch cells, at each cell assembly, are responsible for communicating the occurrence of chunking-out events to their associated map field. The operation of the associative map field is also binary because associative cells are either active or inactive. Similarly, since the associative weights converge to the map field's binary activity pattern, associative weights are also binary.

Associative priming. The associations formed between a pair of cell assemblies can be utilized to send priming signals between $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ in response to chunk-out events. These priming signals are used to increase the competitive strength of the primed cell assembly. This forces that cell assembly to chunk-out its dimension of the pattern in synchrony with the other cell assemblies representing the pattern's other dimensions. However, since multiple associations may exist, multiple cell assemblies may be primed in this way; therefore the set $\mathcal{A}$ will be utilized to determine which cell assemblies should be primed.

Similarly to boundary point alignment, the associative maps described herein are derived from the ARTMAP architecture introduced by Carpenter et al. [21]. Consequently, where appropriate, the terminology will remain consistent. The masking fields $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ are connected via two associative maps — one in each direction, $\mathcal{F}_2^a \rightarrow \mathcal{F}_2^b$ ($\mathcal{F}_2^{ab}$) and $\mathcal{F}_2^b \rightarrow \mathcal{F}_2^a$ ($\mathcal{F}_2^{ba}$). However, since their architecture and operation are identical, their behaviour will be described in terms of the map field from $\mathcal{F}_2^a \rightarrow \mathcal{F}_2^b$ (see Figure 10.8). Additionally, note that, similar to boundary point alignment, associative learning and priming at the phrase level is considered to take place so rapidly that no other network component will evolve during this minute time scale. This is similar to the operation of the ART search process and guarantees that no interference from other network processes will occur.

Map field architecture

For the map field to operate correctly, an additional cell at each cell assembly is introduced (see Figure 10.9). This cell, an associative prime cell labelled a, behaves in exactly the same way as the prime cell described in Section 10.5.1. That is, once activated to unity through chunking-out, it remains active for a brief time period of K_{latch}. Associative prime cells will either cause associative learning or associative priming depending upon the state of the network.

The map field architecture is identical to that presented in Section 10.5.1. $\mathcal{F}_2^{ab}$, which associates $\mathcal{F}_2^a$ with $\mathcal{F}_2^b$, consists of $N_2^{g,b}$ sibling groups each containing $N_2^{s,b}$ siblings (or associative cells), labelled x_{pq}. There is a one-to-many correspondence between cell assemblies at $\mathcal{F}_2^b$ and each map field associative cell (see Figure 10.8). A fixed connection linking a_{pq}^b to x_{pq} communicates the occurrence of a chunk-out event at $\mathcal{F}_2^b$. A reverse connection linking x_{pq} to a_{pq}^b is used to send associative priming signals when synchronizing chunk-outs across all $\mathcal{F}_2$ fields. Each associative prime cell a_{ni}^a at $\mathcal{F}_2^a$ is connected to every $\mathcal{F}_2^{ab}$ associative cell by adjustable excitatory weights, labelled u_{nipq}, which communicate chunking-out event occurrences at $\mathcal{F}_2^a$ to $\mathcal{F}_2^{ab}$. Remember, only the associative map field, $\mathcal{F}_2^{ab}$ is being described here.

Map fields are event-driven

Since these map fields respond directly to chunk-out events they are considered as being event-driven, similar to the map fields described in Section 10.5.1. Chunk-out events affect the the map field differently depending on the state of the network:

Simultaneous chunks-outs at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ cause associative learning. Due to the boundary point alignment process during the segmentation phase, when a chunk-out occurs at one dimension, a search process is initiated at every other dimension to ensure that the

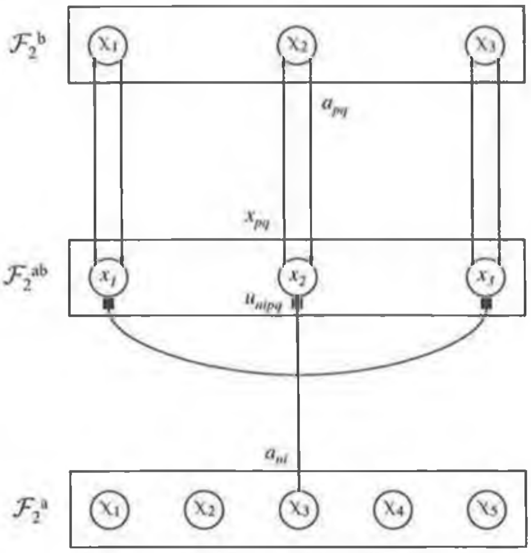


Figure 10.8: The associative map between $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ is used to form permanent associations between cell assemblies (or groups) that represent the pitch sequence and rhythm of the same melodic phrase.

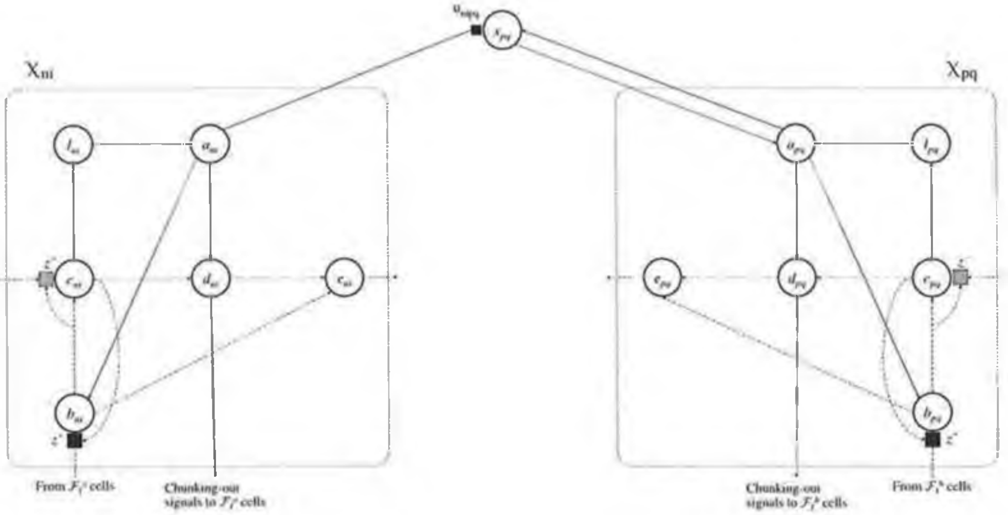

u_{nipq}
x_{pq}
Xni
Xpq
l_{ni}
a_{ni}
c_{ni}
d_{ni}
e_{ni}
b_{ni}
z^*
z^*
a_{pq}
l_{pq}
e_{pq}
d_{pq}
c_{pq}
b_{pq}
z^*
z^*
From $\mathcal{F}_1^a$ cells
Chunking-out signals to $\mathcal{F}_1^a$ cells
Chunking-out signals to $\mathcal{F}_1^b$ cells
From $\mathcal{F}_1^b$ cells

Figure 10.9: An additional cell (a) is used at each cell assembly to facilitate the learning of permanent associations between cell assemblies that respond to the pitch and rhythm dimensions of the same melodic phrase.

entire multidimensional phrase is learned. As a consequence of this search process, each dimension of the phrase will chunk-out at the same time. Latch cells activate in response to chunk-out events at $\mathcal{F}_2$. This in turn causes their corresponding prime cells to activate. When a masking field responds to an event in this way, their corresponding map fields are activated for a brief time period (of K_{latch}) causing associations to form between these active masking fields.

Chunk-out signals from a single $\mathcal{F}_2$ field causes associative priming. The chunking-out of a cell assembly χ_{chunk} at a single $\mathcal{F}_2$ field (say $\mathcal{F}_2^a$) subsequent to the boundary point alignment process will cause the prime cell at χ_{chunk} to activate in response. Prime cells combine with learned associations at $\mathcal{F}_2^{ab}$ in order to attentionally prime any associated cell assemblies at $\mathcal{F}_2^b$. The primed cell assemblies at $\mathcal{F}_2^b$ gain a competitive advantage and may chunk-out their patterns more easily.

Since associative learning and priming are considered to take place so rapidly that no other network component will evolve during this minute time scale, K_{latch} must be sufficiently small to accommodate this. Since associative learning and priming take place on such a small time scale, the chunking-out of other cell assemblies will not interfere with these processes.

Map field cell activations

Each associative cell, x_{pq}, receives input from two sources: $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$. The activation of each associative cell is governed by the following equation (during the time period of K_{latch} after a chunk-out event has occurred):

$$x_{pq} = a_{pq}^b v_c + I_{pq}^{+ab} \tag{10.8}$$

$$I_{pq}^{+ab} = \sum_{i=1}^{N_2^{g,a}} \sum_{n=1}^{N_2^{s,a}} a_{ni}^a u_{nipq} \tag{10.9}$$

where a_{pq}^b is the input on the fixed link from $\mathcal{F}_2^b$. v_c is a fixed weight on this link, which is typically set to 0.1. I_{pq}^{+ab} is the gated input received via the adjustable weights from each associative prime cell, a_{ni}^a, at $\mathcal{F}_2^a$.

Map field learning

The temporal overlap, due to the synchronous chunking-out of cell assemblies at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ (due to boundary point alignment), causes the adaptive pathways of $\mathcal{F}_2^{ab}$ to adjust such that the two cell assemblies become bound. Initially, the adjustable weight from $\mathcal{F}_2^a$ to $\mathcal{F}_2^{ab}$ are set to K_{uinit} (which is typically set to 0.1). Thereafter, the weights are adjusted according to the following (Hebbian-type) learning rule:

$$u_{nipq} = \begin{cases} 1 & \text{if } (a_{ni}^a = 1) \text{ and } (x_{pq} \geq K_{althresh}) \text{ and } (u_{nipq}^{frozen} = false) \\ u_{nipq} & \text{otherwise} \end{cases} \tag{10.10}$$

where $K_{althresh}$ (the associative learning threshold) is the activity an associative cell must reach in order for learning to occur. Typically, $K_{althresh}$ is set to 0.2 $(K_{uinit} + v_c)$. Similarly to χ_{pq}^{com}, individual weights become committed (or frozen) when their adjustable weights reach unity. When this occurs, $u_{nipq}^{frozen} = true$. The third condition ensures that any previous learning is not erased, enabling multiple associations to form.

Note that the same cell assembly, representing a pitch sequence, can be associated with many different rhythmic patterns and vice versa. This one-to-many association is essential, particularly for the current application of segmentation and retrieval. For example, it is common for the same rhythm to be associated with many pitch sequences. Indeed, this illustrates the compact nature of the current method of using separate modules to represent each dimension of an input stream. For instance, using the method proposed by Lewis [88] would mean a separate cell assembly would be required to learn each pitch-rhythm pair. This would cause an unnecessary proliferation of learned phrases at $\mathcal{F}_2$.

Another important issue involves the representation of phrase repetitions at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ (see Section 8.2). According to Equation 10.10, associations are learnt between χ_{ni}^a (a pitch sequence at $\mathcal{F}_2^a$) and χ_{pq}^b (its corresponding rhythm at $\mathcal{F}_2^b$). However, in addition, it is necessary that every cell assembly within χ_{ni}^a's sibling group becomes associated with every cell assembly within χ_{pq}^b's sibling group. Otherwise, due to the manner by which repetition handling operates, priming may not be able to occur between these two fields. This is because there are no guarantees that the two specific cell assemblies, which originally formed the association, will respond to a repeated occurrence of their pattern at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ respectively. In this sense, associations should be considered to form between sibling groups rather that individual cell assemblies. Consequently, identical learning must occur on the adjustable weights connecting every cell assembly within a sibling group at $\mathcal{F}_2^a$ to every associative cell within a sibling group at $\mathcal{F}_2^{ab}$. Since learning should occur identically between these sibling groups, every one of these weight will be the same. This redundancy is similar to the redundancy that exists between the interacting bottom-up links described in Section 6.2.5. For example, suppose $\mathcal{F}_2^a$ contains four siblings in each sibling group and $\mathcal{F}_2^{ab}$ contains three siblings in each group. In this situation, there would be twelve adjustable weight connecting each sibling group from $\mathcal{F}_2^a$ to $\mathcal{F}_2^{ab}$. However, each weight will be identical. Consequently, a single value should be shared between these sibling group pairs. To achieve this, each associative weight is referenced via its originating and terminating groups. For example, the associative weight from the χ_{ni}^a to x_{pq}^{ab} should be references as u_{iq}. This obtains the desired behaviour and has the beneficial side-effect of minimizing computation time.

Activation scenarios

Due to Equations 10.8 and 10.10, the following activation scenarios and responses are possible:

$\mathcal{F}_2^a$ active - $\mathcal{F}_2^b$ active. This situation arises when synchronous chunk-out events have occurred at both $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ due to the boundary point alignment process. Before learning, every associative cell will activate to K_{uinit}, apart from the cell that connects directly to $\mathcal{F}_2^b$, which will activate to $K_{uinit} + v_c$. In this situation, χ^a and χ^b are clearly related. Consequently, associative learning will occur between these assemblies. In the case where learning has already occurred, the appropriate associative cell will activate to $1 + v_c$, while every other associative cell will activate to K_{uinit}.

$\mathcal{F}_2^a$ active - $\mathcal{F}_2^b$ inactive. This situation arises when a chunk-out event has occurred at $\mathcal{F}_2^a$. Provided no learning has occurred, every associative cell at $\mathcal{F}_2^{ab}$ will activate to K_{uinit}. However, if an association has been learnt, a single $\mathcal{F}_2^{ab}$ associative cell will activate to unity. This activation will prime an $\mathcal{F}_2^b$ cell assembly, causing the synchronization of cell assembly chunk-outs.

$\mathcal{F}_2^a$ inactive - $\mathcal{F}_2^b$ active. This situation arises when a chunk-out event has occurred at $\mathcal{F}_2^b$. In this case, a single associative cell will activate to v_c due to the one-to-one fixed links from $\mathcal{F}_1^b$. This occurs regardless of whether learning has occurred or not.

$\mathcal{F}_2^a$ inactive - $\mathcal{F}_2^b$ inactive. In this case, since neither input field is active, no associative cell at $\mathcal{F}_2^{ab}$ will activate.

10.6.2 Synchronizing classifications using associative priming

The associations formed between cell assemblies allow a classification at one module to influence the classification process at another module. This is because associative priming can be utilized. The classification of a pattern at $\mathcal{F}_2^a$ activates the cell assembly's associative prime cell, which in turn causes one or more associative cells, x_{pq}, to fully activate — provided previous associations have been learnt. The activity of x_{pq} is used to prime a_{pq}^b using the following rule:

$$a_{pq}^b = \begin{cases} 1 & \text{if } (G_2 = 1) \text{ and } (x_{pq} = 1) \text{ and } (T_{pq}^b = \mathcal{A}^b \text{ and } M_{pq}^b \geq K_M) \\ 0 & \text{otherwise} \end{cases} \tag{10.11}$$

$$G_2 = \begin{cases} 0 & \text{if } \mathcal{F}_2^a \text{ and } \mathcal{F}_2^b \text{ are both active} \\ 1 & otherwise \end{cases} \tag{10.12}$$

$\mathcal{A}^b$ is utilized to determine whether or not the prime cell should become active or not. As Section 10.5 has already pointed out, the set $\mathcal{A}^b$ will be constructed at $\mathcal{F}_1^b$ in response to a chunk-out at $\mathcal{F}_2^a$. Requiring $T_{pq}^b = \mathcal{A}^b$, enables the classification synchronization process to ensure that both cell assemblies (one at $\mathcal{F}_2^a$ and one at $\mathcal{F}_2^b$) actually respond to the same set of note onsets. This is important because, χ^b's pattern may be present at $\mathcal{F}_1^b$, but in a different context. Furthermore, the match requirement ($M_{pq}^b = K_M$), guarantees that the desired phrase at $\mathcal{F}_2^b$ is synchronized.

Once a_{pq}^b has been primed, it can influence processing in a similar way to how hierarchical priming does (see Section 7.2.2). Cell assemblies that are primed and respond to their complete pattern ($M_{pq} \geq K_M$) are not subject to any chunking-out delay and will therefore, chunk-out their pattern immediately. This associative priming signal ensures that all of the dimensions of a melodic phrase are classified together. Thus, boundary point alignment and binding combine to allow classification synchronization, which enables the formation of a unified phrase representation across $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$. This configuration of rigid associations between parallel modules is advantageous because it enables additional SONNET modules to aggregate sequences of phrase classifications (consisting of pitch-rhythm pairs) forming a hierarchical memory structure that encodes an entire melody. As Chapter 13 will show, it is this binding and hierarchical structure that improves melody recognition, in the form of *Re*TREE*ve*, over previous SONNET research.

Furthermore, the binding of dimensions in this manner allows associative priming. Associative priming enables more robust recognition when presented with incomplete and/or noisy input. For example, suppose the melodic phrase was presented with the pitch dimension being poorly specified, leading to ambiguities over the recognition of the pitch sequence. Nevertheless, the classification of the rhythm can send a priming signal to its corresponding pitch sequence allowing correct classification, thus resolving the ambiguity. Indeed, poor melodic queries in both pitch and rhythm may lead to a situation where mutual associative priming leads to the resolution of ambiguities and correct recognition. Such ambiguities cannot be resolved using only a single melodic dimension.

10.7 Learning multidimensional phrase sequences

In Section 6.4, the manner by which SONNET modules can be cascaded to form a hierarchy was described. In particular, the interactions between the masking field and its adjacent input field in the higher layer was detailed. Apart from the parallel organization of SONNET modules within SONNET-MAP, the interactions and the process of propagating phrase classifications upwards through the hierarchy are identical, and facilitate the learning of two-dimensional phrase sequences (see Figure 10.10).

At the note level, input assemblies at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ (representing pitch categories and IOIs respectively) will activate in response to the same note onset. Similarly, two input assemblies at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ (representing pitch sequence categories and rhythm categories respectively) will activate in response to a multidimensional phrase classification. Multidimensional phrase classifications occur when the separate dimensions of a single melodic phrase are chunked-out at at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ (at approximately the same time) due to the processes of boundary point alignment or classification synchronization. Both chunk-out signals are propagated to their corresponding input assemblies, which in turn activate. This indicates the occurrence of a multi-dimensional phrase occurrence to $\mathcal{F}_3$ (see Figure 10.10). Unlike at the note level, a single classification field is used at $\mathcal{F}_3$. Therefore, $\mathcal{F}_3$ learns sequences of *pitch-rhythm phrase pairings*. Nonetheless, although this is the first time a SONNET module has been configured in this way, the cell activities and adjustable weights evolve in the usual way.

The hierarchical SONNET module, consisting of the input fields at $\mathcal{F}_2^a$ and at $\mathcal{F}_2^b$ and the masking field at $\mathcal{F}_3$, operates in event-driven mode. Furthermore, since $\mathcal{F}_3$ processes pitch-rhythm pairings, the attentional signal governing the network dynamics at this level must be gated by the

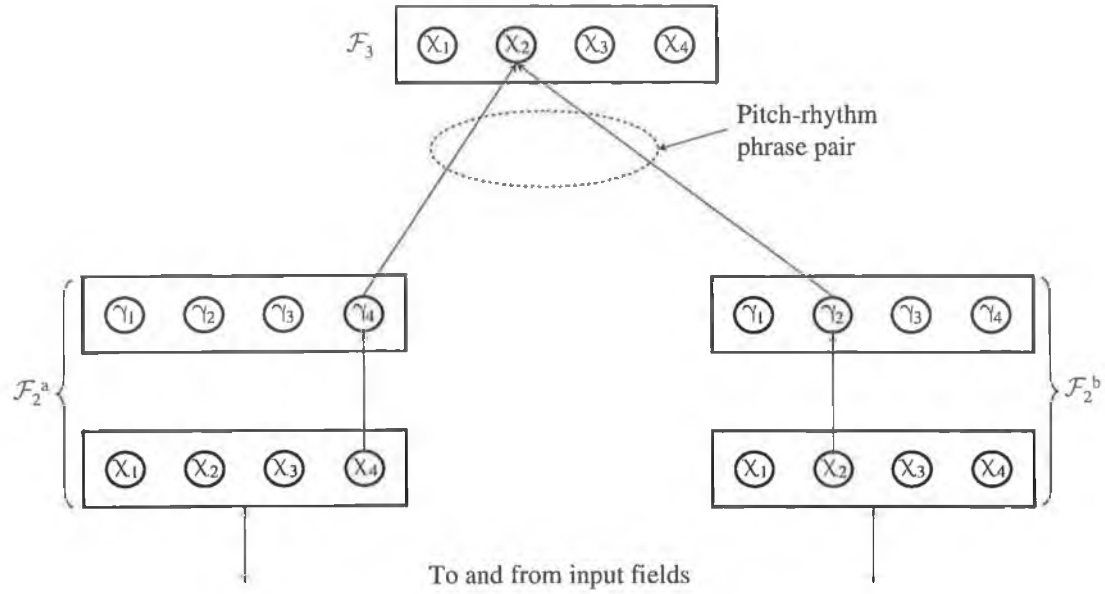


Figure 10.10: The activation of latch cells due to the synchronous classifications at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ cause input cells in the higher layer to activate. The activation of pitch-rhythm pairs in this manner enables $\mathcal{F}_3$ to learn multidimensional phrase sequences.

occurrence of two input events before it activates; one at $\mathcal{F}_2^a$ and one at $\mathcal{F}_2^b$. This ensures that $\mathcal{F}_3$ only ever responds to the classification of pairings.

Allowing phrases consisting of pitch-rhythm pairings to be aggregated in this way enables SONNET-MAP to autonomously learn two-dimensional, parallel and hierarchical representations of entire melodies. This behavioural and organizational configuration is also utilized by the *Re*TREE*ve* network introduced in Chapter 11, which is essential for the recognition process. This is because a form of hierarchical expectation is used to resolve ambiguities in a similar fashion to the way the associative maps described in Section 10.6.2 do. Furthermore, although recall is not the focus of this thesis, Page's work on hierarchical expectation of pitches could be applied to SONNET-MAP and extended to include rhythmic expectation, allowing complete recall of both pitch sequences and rhythms (see Section 14.3).

10.8 Dynamic allocation of resources

Due to the increased scale of the SONNET-MAP architecture, coupled with the potential size of the test suites to be used, the optimization of resources may become an important consideration. With previous SONNET research, the network size could be predicted easily due to the small size of the melody sets used. For example, Page's phrase learning task only consisted of twenty nine phrases. A more difficult test by Roberts [133] showed how rhythmic patterns could be segmented into their constituent phrases. Although Roberts' experiments used ten rhythms ranging from

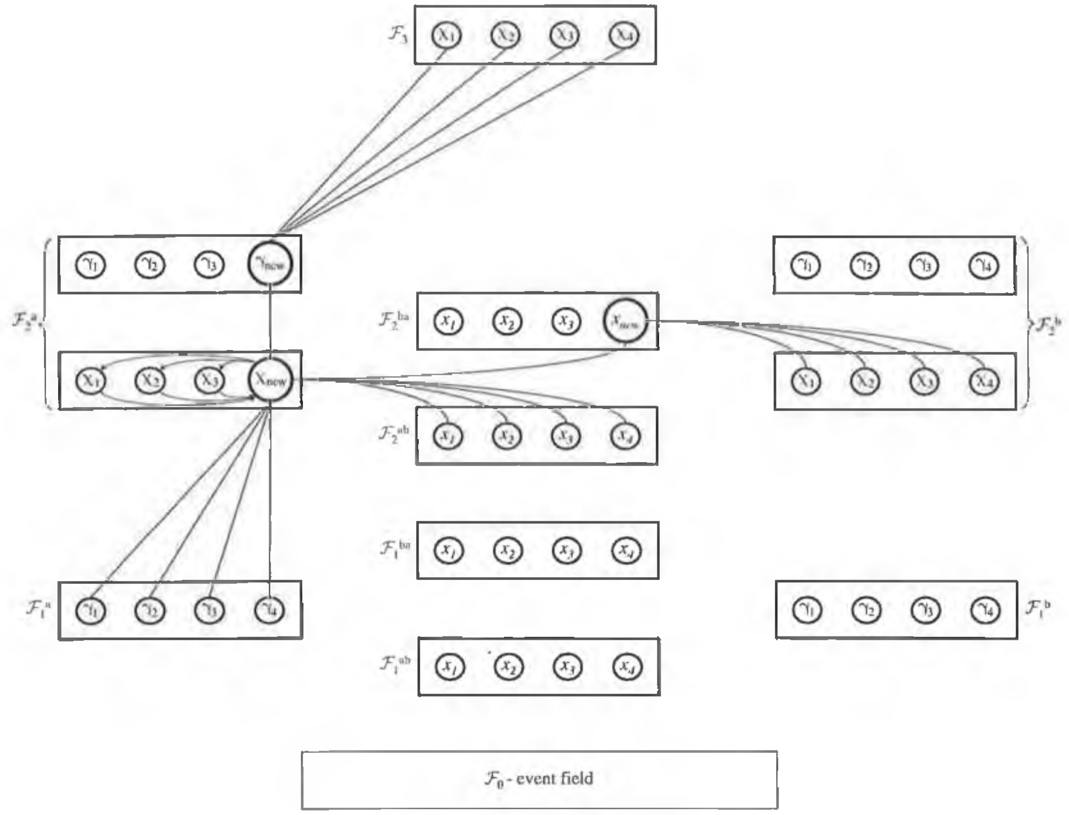


Figure 10.11: The dynamic allocation of resources. A cell assembly group at $\mathcal{F}_2^a$ is appended due to a low cell assembly population. As a result, updates in other areas of the network are also required.

classical to popular music, the network was re-initialized at the end of each rhythm presentation. Consequently, a set of twenty $\mathcal{F}_2$ cell assemblies could reliably learn any of the rhythms presented in these experiments. In this thesis a smimilar approach is taken to segment the *Beatles* test suite.

However, in order to enable SONNET-MAP to process arbitrarily long input sequences, a very large network would be required, which in many circumstances would be excessive. In other words, in real-world situations it may not be possible to predict how large a SONNET-MAP network should be. Instead, a more desirable solution is to enable the network to dynamically determine its own size depending on how learning progresses. This takes considerable pressure off the network designer. To this end, the parameters K_{pop} and K_{growth} are introduced:

- K_{pop} specifies the minimum number of uncommitted cell assembly groups that must always be available to learn.

- K_{growth} specifies the number of cell assembly groups to be allocated when the K_{pop} threshold is breached.

Usually, the number of input assemblies at $\mathcal{F}_1$ can be predetermined and remain fixed. For example, the number of sibling groups (N_1^g) may equal the number of IPCs in Western tonal music, which is twelve (see Chapter 9). The number of siblings within each group (N_1^s) may be commensurate with the STM span of human memory, which is approximately seven. Initially, the number of cell assemblies in the remainder of the network should be relatively small. This enables SONNET-MAP to proceed relatively quickly at first. Furthermore, there should be no need to dynamically allocate any new resources for a reasonable period of time. However, once the K_{pop} threshold has been breached (that is, the number of uncommitted cell assemblies $< K_{pop}$), the additional resources to be allocated is given by K_{growth}. Again, K_{growth} should be large enough that the network can proceed, uninterrupted, for a reasonable time period, but not so large as to unnecessarily slow down the network. In effect, a designer's task is migrated from deciding the exact size of the network towards deciding a strategy on how the network should grow as resource levels become depleted.

When appending a cell assembly (typically at $\mathcal{F}_2^a$ or $\mathcal{F}_2^b$), due to the low resources threshold specified by K_{pop}, the following resource allocation ensues (in this case, it is assumed that a cell assembly at $\mathcal{F}_2^a$ is appended) (see Figure 10.11):

- A cell assembly (χ_{new}) is appended to the masking field that is running low on resources. In this case, at $\mathcal{F}_2^a$.

- The corresponding input assembly (γ_{new}) at $\mathcal{F}_2^a$ is also appended. This ensures that any new phrase categories can be communicated to $\mathcal{F}_3$ when forming hierarchical phrase sequences. Furthermore, fixed one-to-one connections in both directions need to be added. This facilitates communication between χ_{new} and γ_{new}.

- An additional associative assembly (x_{new}) is also required at $\mathcal{F}_2^{ba}$. Furthermore, fixed one-to-one connections in both directions need to be added to facilitate communication between χ_{new} and x_{new}.

As a consequence of the allocation of these cell resources, additional connections are also required:

- Excitatory bottom-up connections from each $\mathcal{F}_1^a$ input assembly to χ_{new} are required. Furthermore, reciprocal feedback connections from χ_{new} to every $\mathcal{F}_1^a$ input assembly are also required.

- Excitatory bottom-up connections from γ_{new} to every cell assembly at $\mathcal{F}_3$ are required. Furthermore, reciprocal feedback connections from $\mathcal{F}_3$ to γ_{new} are also required.

- Adjustable associative connections from every $\mathcal{F}_2^{ab}$ cell to χ_{new} are required.

- Adjustable associative connections from every $\mathcal{F}_2^b$ cell assembly to x_{new} are required.

Using this scheme, preliminary simulations showed a significant improvement in the performance of the network, particularly at its initial operational stage.

10.9 Summary

Chapter 9 showed how each dimension of an input stimulus could be represented by distinct and parallel input fields ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$), representing SONNET-MAP's working memory. This chapter extended this idea by showing how parallel SONNET modules can be associated at the note level and at the phrase level to form parallel and hierarchical melody encodings. Since this is the first time a SONNET-based neural network has been used to automatically segment a multi-dimensional input stimulus using a parallel neural architecture, the new techniques of boundary point alignment and classification synchronization were introduced. Page showed the benefits of using a SONNET hierarchy over a single SONNET module. The coming chapters will show the benefits of using SONNET-MAP over the use of single dimensional SONNET hierarchies.

Chapter 11

*Re*TREE*ve*

11.1 Introduction

Chapters 10 and 12 introduce SONNET-MAP and show how its extended architectural configuration enables both pitch and rhythm to be utilized in discovering phrase boundaries within a melody. Pitch sequence and rhythm categories are formed (using information at the note level) and represented independently at separate parallel modules. Related pitch and rhythm sequences are bound using associative maps, forming multidimensional phrase representations. However, problems exists when using SONNET-MAP in its current form as a CBR system for melodies. The traditional *winner-take-all* (WTA) competitive strategy does not fare well when applied to CBR, because global constraints cannot be used to aid in the recognition process. Furthermore, SONNET-MAP's operation does not scale well to large data sets, particularly when implemented on serial computers. This is primarily due to the differential equations used to model the passage of time.

Consequently, this chapter introduces a new network, known as *Re*TREE*ve*, which adopts many aspects of the SONNET-MAP network (particularly its novel organizational structure) and addresses the issues relating to competition and performance. These modifications enable *Re*TREE*ve* to be used as an efficient, robust and accurate CBR system. Note that some of the symbols used to represent certain components of this network may overlap with previous chapters. Therefore, to avoid confusion, it is best to consider these components independently of previous work, although some of the concepts behind them may be similar or identical.

11.2 Applying SONNET-MAP to the CBR of melodies

This section illustrates the problems associated with using SONNET-MAP as a retrieval system.

11.2.1 Business as usual

An obvious approach to using SONNET-MAP as a CBR system for melodies is to allow weights and cell activations to evolve in the usual way. Then, once a collection of melodies have been learned, queries can be presented to SONNET-MAP in the same way as a normal melody would. The cell assembly (and its corresponding melody) with the largest activation at the most abstract layer in the network would constitute the retrieved melody in response to a query. However, this approach proved problematic during preliminary simulations for the following reasons:

- Previous SONNET-based research has dealt with relatively small data sets [114, 118, 133]. These data sets served their purpose in illustrating SONNET's operation at the task in question (for example, melody recall and rhythm perception). However, when the task at hand is CBR, large data sets are common. Therefore, scalability and performance considerations need to be taken into account.
- So far, SONNET-based research has focussed on the problems of segmentation, learning and recall. The desired behaviour was achieved using WTA competitive fields at $\mathcal{F}_2$, which only required a local context to function as intended. However, when the task is that of CBR, global constraints need to to be satisfied too. Section 11.2.3 shows that WTA competition is too restrictive for CBR.

These issues are discussed in more detail in the remainder of this section. Following that, the manner in which these problems are resolved using *Re*TREE*ve* are introduced.

11.2.2 Performance considerations

Human ability to process, organize and recognize the multifaceted features of an ever-evolving musical pattern is vastly superior than the performance achieved by any serial machine produced thus far. When one considers that the response time of a neuron is relatively slow when pitted against modern digital sequential processors, one can appreciate the advantages gained from parallel architectures. The simulation of any inherently parallel architecture on a sequential processor restricts the scale of the problems that can be tackled by that system. This is particularly true for neural networks due to the granularity at which parallelism operates. Thus, the performance gain enjoyed by parallel architectures somewhat evaporates when they are simulated on serial machines. This point is further reinforced by Page [118, pg. 317] in relation to SONNET:

> *'The intrinsically "parallel" nature of many of the calculations performed during simulation, suggests that distribution of the computational load across many parallel processors might provide a solution to the problem of excessive run-time'*

Moreover, when one is trying to process melodies, the complexity of modelling time comes to the fore. The evolution of classification cells and adjustable weights must be specified in terms of differential equations that are integrated at sufficiently small time intervals (at most 2ms) so as to model the passage of time in a stable manner (refer to Section D.5). This places a considerable strain on computation time. These problems are further exacerbated by SONNET-MAP's multidimensional (and potentially hierarchical) configuration. In fact, the simulations reported in Chapters 12 and 13 represent the largest test of SONNET-based neural networks reported to date. This includes its architectural specification, such as the quantity of input cells, classification cells and connections, as well as its application to a difficult real-world problem. Nevertheless, since massively parallel architectures are not commonplace, and in order to apply neural network technology in the real-world, compromises must be made in order to produce industrial strength applications.

11.2.3 Suitability of winner-take-all competition for melody retrieval

In general, SONNET uses a WTA competitive strategy, with one caveat: if the winners are disjoint, SONNET does allow multiple winners (see Section 5.3.6). The utilization of a WTA strategy for implementing competition at $\mathcal{F}_2$ has, thus far, served SONNET-based research well. Using this scheme, phrase boundaries in a melody can be discovered and learned. Moreover, feedback can be used to bias expectations and recall entire melodies. However, when the problem is to retrieve a melody based on an input query, a WTA competitive strategy is too restrictive. To illustrate this point, consider the following example with reference to Figure 11.1.

Suppose *Melody 1* and *Melody 2* have been learned at $\mathcal{F}_3$. Furthermore, suppose a query is presented at $\mathcal{F}_1$ and that this query originates from *Melody 1*. When the fourth note (E) is presented, the phrases {B, C, D, E} and {C, D, E} will be recognized by their respective cell assemblies, χ_{BCDE} and χ_{CDE}. However, due to the WTA competitive strategy of SONNET, only one of these cell assemblies will perform a classification, and chunk-out its pattern from $\mathcal{F}_1$. This in turn will trigger the activation of its corresponding latch cell, which communicates the occurrence of its classification to $\mathcal{F}_3$. Due to the competitive strength that larger cell assemblies have over smaller cell assemblies (see Section 5.3.6), χ_{BCDE} will win this competition, and at this point, will lead to the retrieval of *Melody 2*. Due to the chunking-out of χ_{BCDE}, the occurrence of the phrase {C, D, E} will never be known to $\mathcal{F}_3$.

On presentation of the eight note B, the phrase {F, G, A, B} will be recognized and classified by χ_{FGAB}. Although the classification of {F, G, A, B} will contribute to the retrieval process at $\mathcal{F}_3$, *Melody 1* will never be retrieved ahead of *Melody 2*. This is because it is connected to χ_{FGAB} by a weaker excitatory weight than the connection from χ_{BCDE} to *Melody 2*. This is due to the manner in which excitatory weights evolve towards a decreasing activation pattern (see Section 5.3.1).

This scenario points out SONNET's inability to behave in a manner that allows the satisfaction of global, context-sensitive constraints. In other words, the winner at one particular point in time does not necessarily emerge as the winner once the global context emerges. Section 11.5 will show how {B, C, D, E}, {C, D, E} and {F, G, A, B} can all contribute to the retrieval process at $\mathcal{F}_3$. This will lead to the recognition of *Melody 1* as the query's source, thus achieving correct recognition and overcoming this problem.

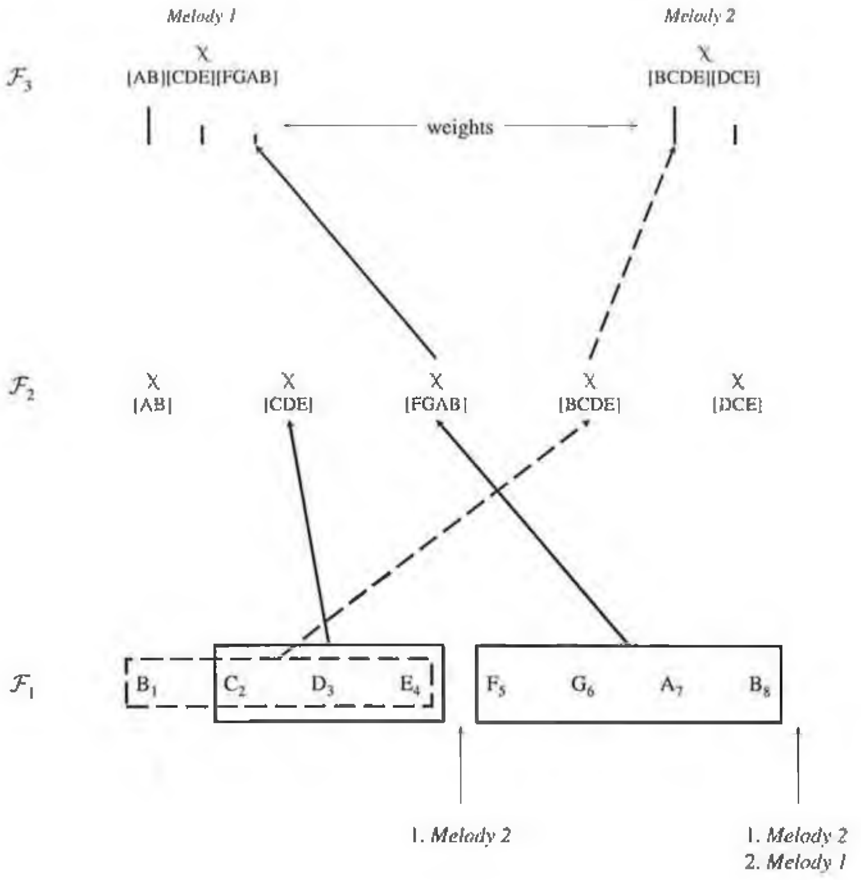


Figure 11.1: A WTA strategy fails to allow both χ_{CDE} and χ_{FGAB} to contribute to the correct retrieval of *Melody 1*. For simplicy, the use of a single SONNET hierarchy is used to illustrate this problem.

11.3 Introducing *Re*TREE*ve*

In order to address the issues raised above, this chapter introduces a new retrieval architecture and algorithm known as *Re*TREE*ve*, which adopts many aspects of the SONNET-MAP network (particularly its novel organizational structure). This structure, and the modifications introduced in this chapter, enable *Re*TREE*ve* to be used as an efficient, robust and accurate CBR system. The architectural configuration and LTM structure for *Re*TREE*ve* is specified in this section.

11.3.1 Architecture

- $\mathcal{F}_0$ is divided into a field containing 88 cells labelled e_l, each responding to a particular CPH spanning the usable pitch range of human hearing. Communication between $\mathcal{F}_0$ and each input field is achieved using the appropriate gating mechanism (see Chapter 9).

- Each input field ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) consists of N_g sibling groups. Each sibling group contains N_s siblings labelled s_{mj}, where j indexes a specific sibling group and m indexes a specific sibling within that group. In addition to the activation of each cell, the event number of the note which caused the cell to activate is also recorded. For example, the event number associated with the tenth note of a query is 10. The structure of the input fields of *Re*TREE*ve* are identical to those of SONNET-MAP.

- Each match field ($\mathcal{F}_2^a$ and $\mathcal{F}_2^b$) consists of N_2 match nodes labelled m_i. This is a marked simplification when compared against SONNET-MAP's $\mathcal{F}_2$ architecture. In particular, no sibling group architecture is required and no inhibitory or feedback connections exist. Initially, every match node is considered to be connected to every input cell at its corresponding input field. Furthermore, the excitatory weight on each connection is initialized to zero. Consequently, multiple links and presynaptic inhibition need not be used. Instead, pattern tracking (repetition handling) is achieved by ensuring that all match nodes always respond to the most recent note onsets at their corresponding input field.

- The template field at $\mathcal{F}_3$ consists of N_3 template nodes labelled t_k. Again, this is a marked simplification when compared against SONNET-MAP's $\mathcal{F}_3$ architecture — no sibling group architecture, no presynaptic inhibition, no inhibitory connections and no feedback connections. However, unlike at the match fields, no pattern tracking needs to occur. Furthermore, every template node is considered to be connected to every match cell at both $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$. Each connection has a fixed weight of unity, and their only function is the communicate the occurrence of each pitch-rhythm pair at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ that match their input patterns at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ respectively.

11.3.2 Long-term memory structures

*Re*TREE*ve*'s LTM contains three data structures (analogous to those of SONNET-MAP's) which are populated during learning and utilized during retrieval.

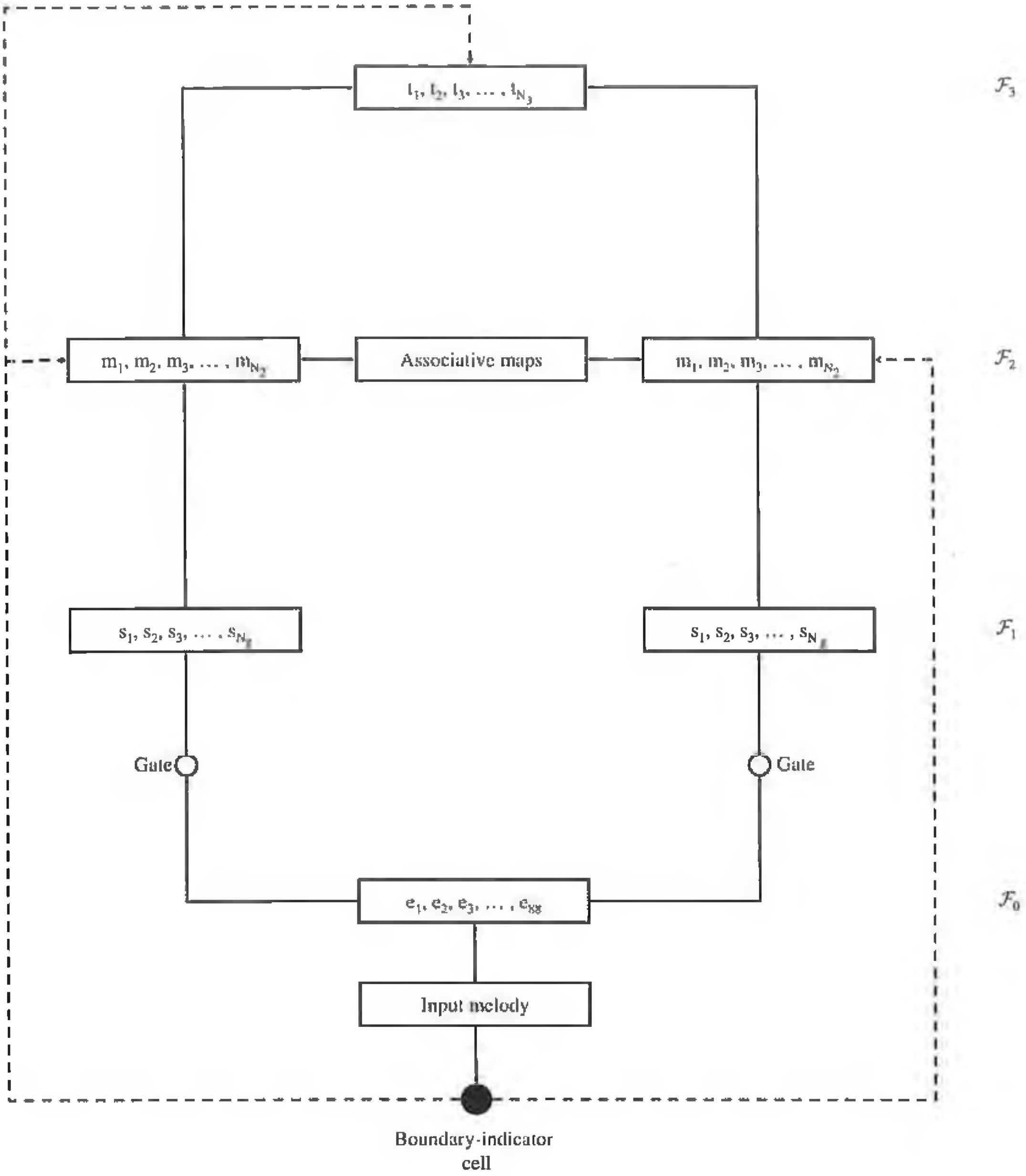

$t_1, t_2, t_3, \ldots, t_{N_3}$
$\mathcal{F}_3$
$m_1, m_2, m_3, \ldots, m_{N_2}$
Associative maps
$m_1, m_2, m_3, \ldots, m_{N_2}$
$\mathcal{F}_2$
$s_1, s_2, s_3, \ldots, s_{N_1}$
$s_1, s_2, s_3, \ldots, s_{N_1}$
$\mathcal{F}_1$
Gate
Gate
$e_1, e_2, e_3, \ldots, e_{88}$
$\mathcal{F}_0$
Input melody
Boundary-indicator cell

Figure 11.2: *Re*TREE*ve* CBR system.

The learned weights at $\mathcal{F}_2$. Each match node, m_i, contains a set T_i, which consists of the set of indices that identify the specific sibling groups at $\mathcal{F}_1$ that constitute m_i's learned pattern. Each index in T_i has a corresponding excitatory weight, z^+_{ji}, which encodes the content of that pattern. T_i is specified in descending order of excitatory weight strength. Phrase tracking is achieved by ensuring that T_i always responds to the $|T_i|$ most recent note onsets occurring at its corresponding input field.

The learned associations at $\mathcal{F}_2$. Each match node, m_i, contains a set of indices specified by $\mathcal{A}_i$, which identify the match nodes, m_j, that they are associated with. That is, each pitch sequence has a list of associated rhythms and vice versa. This models the associations formed between the classification fields of the SONNET-MAP network.

The learned melody templates at $\mathcal{F}_3$. The set $\mathcal{T}_k$ at each $\mathcal{F}_3$ template node, t_k, specifies all of the pitch-rhythm phrase pairings of a particular melody. That is, the set of index pairs (representing $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ match nodes) which encode the phrase sequence of a melody. In addition, the order in which these these pairings are expected to occur is also specified. In this regard, $\mathcal{T}_k$ represented a melody's phrase template, which is matched against during the retrieval process. The set $\mathcal{T}_k$ is constructed during the time *Re*TREE*ve* is being populated with melodies. The first member of each pair denotes the i^{th} pitch sequence represented by m_i at $\mathcal{F}_2^a$, while the second member denotes the j^{th} rhythm represented by m_j at $\mathcal{F}_2^b$ (see Figure 11.3).

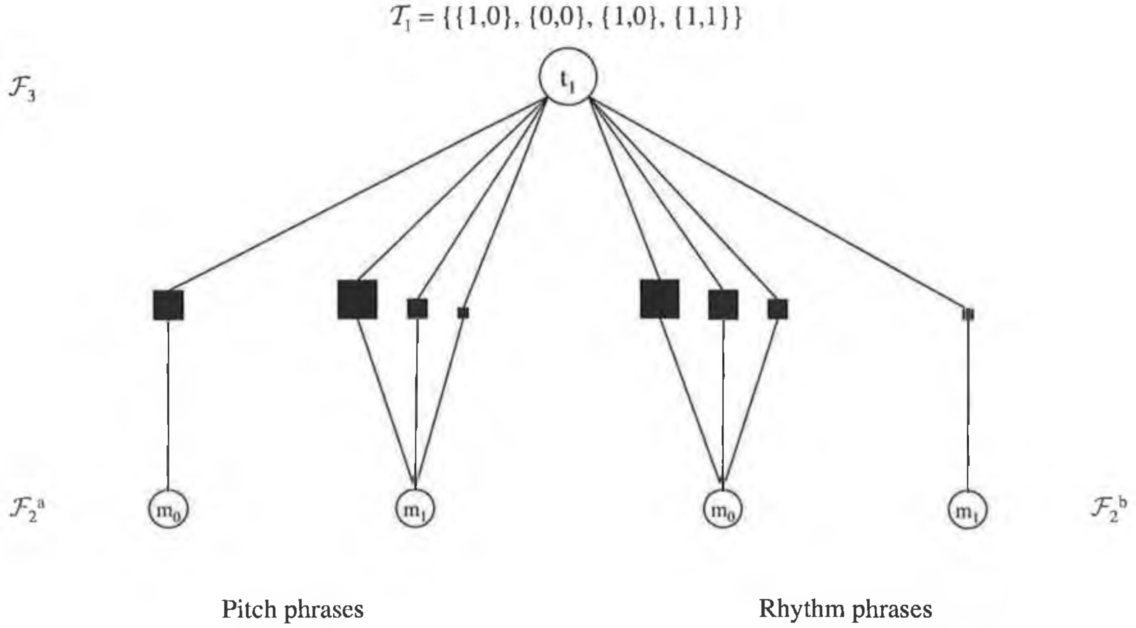


Figure 11.3: The set $\mathcal{T}_k$ specifies each pitch-rhythm phrase pairing of a melody. In addition, the order in which these pairings are expected to occur is also specified. In this regard, $\mathcal{T}_k$ represents a melody's phrase template, which should be matched against during the retrieval process.

11.4 Populating *Re*TREE*ve* with melodies

Before *Re*TREE*ve* can be used for retrieval purposes, it must first be populated with melodies. Chapters 10 and 12 show how SONNET-MAP can be utilized to discover boundary points in a melody. These boundary points are recorded in the original melody files presented to SONNET-MAP. Then, each phrase which is braced by a pair of boundary points is learned directly by *Re*TREE*ve*. This is achieved through the use of a boundary-indicator cell, which fires in response to the boundary point representing the end of each phrase. Note also, since *Re*TREE*ve* learns pre-segmented input sequences, the melody files do not need to have been created by SONNET-MAP. This allows other segmentation regimes to operate, which would be of particular value when comparing the retrieval performance of *Re*TREE*ve* using SONNET-MAP segmentations with the retrieval performance of *Re*TREE*ve* using other regimes.

11.4.1 Boundary-indicator cell

Section 10.5 described the process of boundary point alignment, which occurs when a cell assembly in either the pitch or rhythm dimensions becomes committed to a pattern. This section describes a similar process, whereby a boundary-indicator cell is activated in response to a boundary point indicating the end of a phrase. When the boundary-indicator cell is fired, one or both of the following happens:

- Each match node is checked to see if it has already learned the pattern.
- If the pattern has not previously been learned, the next available match node will be utilized to learn the current pattern at $\mathcal{F}_1$.

11.4.2 Learning each phrase at $\mathcal{F}_2$

To ensure that only a single match node learns each phrase of a melody presented to $\mathcal{F}_1$, the activation (m_i) of each match cell must be computed. m_i is the only value required to quantify the match between a cell assembly's LTM weight vector and its STM input pattern. This value is computed similarly to SONNET-MAP's match value, M_i. Although minor changes as to how this quantity is usually calculated are introduced here, the concept of a match between a cell assembly's excitatory weights and its pattern in working memory remains identical. m_i only needs to be calculated once per event onset, and is computed as follow:

$$m_i = \frac{I_i^*}{|T_i|} \tag{11.1}$$

$$I_i^* = \sum_{j \in T_i} I_{mji}^\times \tag{11.2}$$

$$I_{mji}^\times = \min\left(\frac{Z_{mji}}{S_{mji}}, \frac{S_{mji}}{Z_{mji}}\right) \tag{11.3}$$

$$Z_{mji} = \frac{z_{mji}^+}{\left[\sum_{j \in T_i} (z_{mji}^+)^2\right]^{1/2}} \tag{11.4}$$

$$S_{mji} = \frac{s_{mji}}{\left[\sum_{j \in T_i} (s_{mji})^2\right]^{1/2}} \tag{11.5}$$

Although this formulation is similar to that presented in Section 5.3, $I_i^\times$ is removed from the formulation. This is due to a shift away from the WTA competitive strategy employed thus far, towards multiple winners. This slimmed down version of m_i reduces computation time considerably. However, the primary reduction in computation time is due to fact that no cell activations or weights need to evolve at any match node during learning and retrieval. A node at $\mathcal{F}_2$ is considered to match its pattern when m_i reaches its match threshold K_M ($m_i \geq K_M$). For learning, K_M should be set fairly high because the goal is to perfectly learn each phrase of every melody.

If $m_i \geq K_M$ and $|T_i|$ is equal to the number of active input assemblies at $\mathcal{F}_1$, no further action is required because the pattern at $\mathcal{F}_1$ has already been learned. However, if no match cell satisfies these conditions, the phrase at $\mathcal{F}_1$ must be learned by an un-instantiated match node. Learning the input pattern at $\mathcal{F}_1$ is simply a matter of constructing the set T_i at m_i. To achieve this, the weight on every connection to each input cell at $\mathcal{F}_1$ is set to the normalized input of that cell, which is represented by S_{mj}. Each S_{mj} is calculated as follows:

$$S_{mj} = \frac{s_{mj}}{\left[\sum_{m,j \in \mathcal{F}_1} (s_{mj})^2\right]^{1/2}} \tag{11.6}$$

Furthermore, if $S_{mj} > 0$, its associated group becomes a member of T_i. At this point, $m_i \geq K_M$ at the newly instantiated match node.

After each phrase has been presented and learned, the entire input field is reset for the forthcoming phrase. However, if the last onset of a phrase represents the beginning of a subsequent phrase (which will be indicated by the boundary point recoded in the melody input file), this onset will not be reset.

11.4.3 Learning associations between phrases at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$

Binding must take place between match nodes that represent different dimensions of the same melodic phrase, in response to the commitment process described above (see Section 10.6.1). It is imperative that these associations are learned because they are vital for the CBR algorithm described in Section 11.5. When a boundary point occurs in a melody and the phrase at this boundary point has been learned, there will be two match nodes m_i and m_j (at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ respectively) which will have reached their match thresholds. Furthermore, the length of the pattern they encode will equal the length of the phrase just presented. In this situation, m_i will add the index j to $\mathcal{A}_i^a$. Conversely, m_j will add the index i to $\mathcal{A}_j^b$. Furthermore, no duplicates are allowed in the sets $\mathcal{A}_i^a$ and $\mathcal{A}_j^b$ in response to multiple occurrence of the same input pattern.

11.4.4 Learning the phrase sequence templates at $\mathcal{F}_3$

Chapters 6 and 10 showed how the hierarchical structure of a melody can be discovered by cascading multiple SONNET modules. However, if the task at hand is to recognize sequences of notes that form phrases and the order in which these phrases occur, there is no need to employ more than one additional module. Consequently, $\mathcal{F}_3$ learns the entire phrase sequence of every melody.

Again, when a boundary point occurs in a melody, there will be at most two match nodes m_i and m_j (at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ respectively) which reach their match thresholds K_M^a and K_M^b and whose pattern lengths equals the length of the phrase currently presented. In this circumstance, the indices i and j form a pair which will be added to the set $\mathcal{T}_k$ at $\mathcal{F}_3$. Each melody will be stored using a separate template node at $\mathcal{F}_3$.

11.5 Retrieving melodies

Section 11.4 showed how the melodies segmented using SONNET-MAP can be used to populate the *Re*TREE*ve* network. The task of pattern recognition faces different constraints than those of pattern segmentation and learning. Furthermore, in light of the need to perform as efficiently as possible, the operation of *Re*TREE*ve* is markedly simplified compared to SONNET-MAP, and yet remains robust when applied to pattern recognition. This section shows how information encoded in the *Re*TREE*ve* network can be utilized to retrieve melodies.

11.5.1 Resetting $\mathcal{F}_1$ cells

The activity of the input cells operate in exactly the same manner as described previously, with one exception. Every match node at $\mathcal{F}_2$ only responds to the $|T_i|$ most recent note onsets at $\mathcal{F}_1$. Furthermore, the maximum size of this set is five; $|T_i| = N_l = 5$. Therefore, only the five most recent events need to be stored at $\mathcal{F}_1$. Consequently, input cells which are older than the five most recently activated input cells are reset.

11.5.2 Phrase matching and multiple winners at $\mathcal{F}_2$

m_i (see Equation 11.1) is the only value required to quantify the match between a cell assembly's LTM weight vector and its STM input pattern. A node at $\mathcal{F}_2$ considered to match its pattern when m_i reaches the match threshold K_M, which can be specified so as to enable noisy input to be processed. Therefore, all match nodes that reach their match threshold will influence recognition at $\mathcal{F}_3$. In other words multiple winners are allowed, which resolves the issues discussed in Section 11.2.3. m_i is calculated when each note of a query is presented to *Re*TREE*ve*.

11.5.3 Specifying associated pitch-rhythm phrase pairs

Thus far, this chapter has described how pitch phrases and rhythm phrases are recognized independently by their respective modules. This section shows how complete multidimensional phrases (that is, a pitch sequence and its associated rhythm) can be recognized as an amalgam of two independent dimensions. As Chapter 13 will show, a single dimensional hierarchy retrieves melodies more poorly than a two dimensional hierarchy. When pitch and rhythm information is available, only associated pitch-rhythm phrase pairs are allowed to contribute towards the recognition of a melody at $\mathcal{F}_3$. Therefore, recognition at $\mathcal{F}_3$ is not thwarted by the proliferation of unwarranted single dimensional phrases. Indeed, Chapter 13 shows that if either of the dimensions of pitch or rhythm is poorly specified relative to the other, the associations that exist between the pitch phrase and the rhythm phrase can be used to resolve any ambiguity, which enhances the retrieval results. This is not possible with systems that use a single dimension. Associated pitch-rhythm pairs are recognized using the associations formed between pitch sequences and their corresponding rhythms, which are specified by the sets $\mathcal{A}_i^a$ and $\mathcal{A}_j^b$. This section shows how these associations can be utilized to propagate associated pitch-rhythm pairings to $\mathcal{F}_3$.

To achieve this, the set $\mathcal{P}$ is introduced, which is constructed at $\mathcal{F}_2$. The set $\mathcal{P}$ is used during retrieval to yield a match at each template node at $\mathcal{F}_3$. $\mathcal{P}$ contains a list of the pitch sequence-rhythm pairs that match the current input query at $\mathcal{F}_1$. The i^{th} node at $\mathcal{F}_2^a$ and the j^{th} node at $\mathcal{F}_2^b$ are considered to be a matched pair (and consequently, members of $\mathcal{P}$) if the following conditions are met:

- m_i^a has a learned association with m_j^b, that is if $j \in A_i$ (or vice versa).
- If m_i^a and m_j^b both match their corresponding input patterns at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ respectively. That is $m_i^a \geq K_M^a$ and $m_j^b \geq K_M^b$.
- If m_i^a and m_j^b are aligned at the event level. This can be determined if $|T_i^a| = |T_j^b|$.

Mathematically, a template node at $\mathcal{F}_3$ can determine if a pitch-rhythm pair specified in its template $\mathcal{T}_k$ is member of $\mathcal{P}$:

$$\mathcal{T}_{k_{ij}} \in \mathcal{P} \iff (m_i^a \geq K_M^a) \text{ and } (m_j^b \geq K_M^b) \text{ and } (m_j^b \in A_i^a) \text{ and } (|T_i^a| = |T_j^b|) \tag{11.7}$$

In effect $\mathcal{T}_k$ deals with expected matched pairs, whereas $\mathcal{P}$ deals with actual matched pairs that currently exist, based on the current input query. Note also, associated with each matched pair is a set of indices that identify the event numbers of the notes which resulted in the recognition of the matched pitch-rhythm pair. This set is labelled $\mathcal{E}$.

11.5.4 Computing the ranked list at $\mathcal{F}_3$

Each template node, t_k, at $\mathcal{F}_3$ matches the phrase pairs specified by $\mathcal{P}$ against its melody template $\mathcal{T}_k$. The length of the largest phrase sequence, $\mathcal{P}_{seq}$, consistent with $\mathcal{T}_k$ constitutes t_k's match value. Adjacent elements of the set $\mathcal{P}_{seq}$ must be aligned. A pair of adjacent elements are considered to be aligned if:

1. They occur in the correct order with respect to $\mathcal{T}_k$.

2. The note onsets of each element are contiguous. This can be determined by examining the event numbers specified by each pitch-rhythm pair (that is, $\mathcal{E}$). To enable a certain degree of overlap, which is required by interval representations, note onsets may overlap by a certain amount. The degree of overlap allowed is specified by the parameter $K_{overlap}$, which is generally set to 1. Remember, for interval encoding schemes, the last onset of one pattern may represent the beginning of a subsequent pattern.

The length of $\mathcal{P}_{seq}$ is measured in terms of unique onset occurrences, represented by each phrase pair in the set. Consequently, a phrase sequence consisting of two phrases may give a match value greater than a phrase sequence consisting of three phrases, depending on the size of the phrases in each sequence. It is these match values that are used to produce a descending list of matches known as a ranked list. In light of the work presented so far, the match value for each template node at $\mathcal{F}_3$ is specified by Algorithm 11.1.

Due to restrictions imposed on the number of pitch-rhythm pairs allowed to be considered for matching at each $\mathcal{F}_3$ node, the search space has been significantly reduced. However, Algorithm 11.1 performs an exhaustive search in which every $\mathcal{F}_3$ node is considered. Consequently, scaling issues need to be examined as the number of melodies stored at $\mathcal{F}_3$ increases. Although these issues are not addressed in this thesis, we do believe that improvements can be made to Algorithm 11.1, whether implemented on a serial or parallel computer, which may further restrict the search space; thus affording the algorithm greater scalability. One possibility would be to classify the genre of each query and to only search the $\mathcal{F}_3$ nodes that are within that genre.

This algorithm can be explained with reference to Figure 11.4. To compute the match value at an $\mathcal{F}_3$ template node, each pitch-rhythm pair in the template $\mathcal{T}$ will be matched against the actual pitch-rhythm recognitions that have been performed at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ and specified by $\mathcal{P}$. In this example,

$$\begin{aligned} \mathcal{T} &= \{\{0,2\},\{1,3\},\{2,4\},\{3,5\},\{4,6\},\{5,9\}\} \\ \mathcal{P} &= \{\{1,3\},\{4,6\},\{2,4\},\{0,2\},\{3,5\},\{5,9\}\} \end{aligned}$$

Now, consider the following lines of enquiry (LOE) with respect to Algorithm 11.1:

LOE 1: The first pair specified by $\mathcal{T}$ is $\{0, 2\}$. Therefore, the phrase encoded by m_0^a and m_2^b represents the first pattern that is expected to occur. At this point $t_{pair} = \mathcal{T}[1]$ (see line 4 of Algorithm 11.1). In this example, $\{m_0^a, m_2^b\}$ do in fact form a matched pair because

Algorithm 11.1 The *Re*TREE*ve* algorithm.

```
procedure ComputeMatch( $\mathcal{T}$ )
    $match = 0$
    for $n = 1$ to $|\mathcal{T}|$ do
        $t_{pair} = \mathcal{T}[n]$
        $\mathcal{P}_{sub} = \mathcal{P}[t_{pair}]$
        for $m = 1$ to $|\mathcal{P}_{sub}|$ do
            $p = \mathcal{P}_{sub}[m]$
            $currentMatch = SequencePhrases(p,\ \mathcal{T}, n + 1)$
            if $currentMatch > match$ then
                $match = currentMatch$
            end if
        end for
    end for
    return $match$
end procedure

procedure SequencePhrases( $p$, $\mathcal{T}$, n )
    $pairMatch = |p.\mathcal{E}|$
    $seqMatch = 0$
    if $n \neq |\mathcal{T}|$ then
        $t_{pair} = \mathcal{T}[n]$
        $\mathcal{P}_{sub} = \mathcal{P}[t_{pair}]$
        for $m = 1$ to $|\mathcal{P}_{sub}|$ do
            $p_{next} = \mathcal{P}_{sub}[m]$
            if $AreContiguous(p, p_{next})$ == true then
                $currentMatch = SequencePhrases(p_{next},\ \mathcal{T}, n + 1)$
                if $currentMatch > seqMatch$ then
                    $seqMatch = currentMatch$
                end if
            end if
        end for
    end if
    return $pairMatch + seqMatch$
end procedure
```

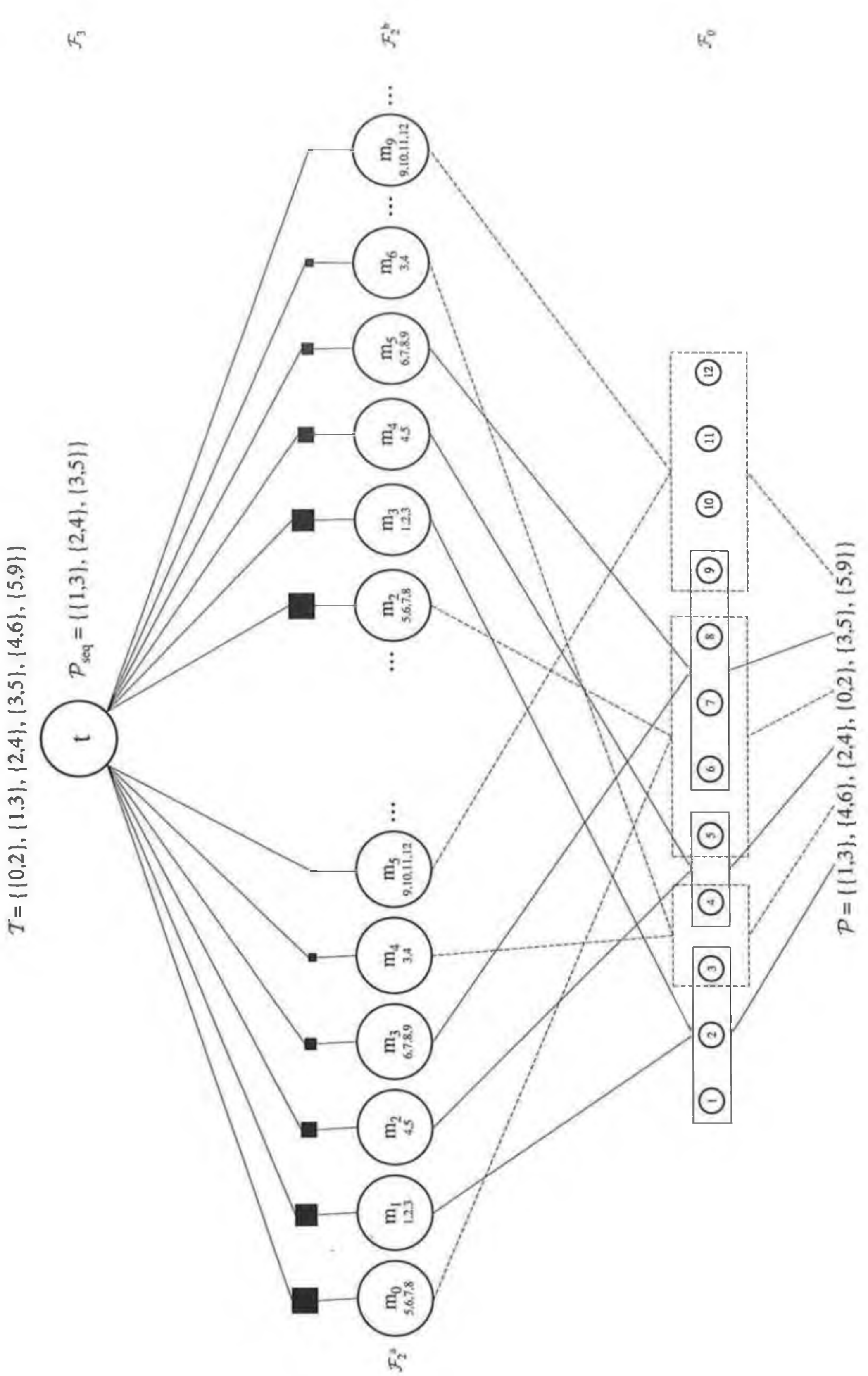


Figure 11.4: The operation of the *Re*TREE*ve* CBR algorithm. In this diagram, a single template cell, t, is illustrated along with its melody template $\mathcal{T}$. For simplicity, the twelve notes that have occurred in this example are indicated by their event numbers at $\mathcal{F}_0$. Furthermore, the current list of matched pairs ($\mathcal{P}$) is also specified with their associated sets $\mathcal{E}$ indicated at $\mathcal{F}_0$.

their input is present and they are elements of $\mathcal{P}$. At this point, $\mathcal{P}_{sub} = \{\{0, 2\}\}$ (see line 5 of Algorithm 11.1). Note that since many occurrences of the same matched pair may occur in different contexts within an input query, the set $\mathcal{P}_{sub}$ may contain many matched pairs representing the same pattern. In this example $|\mathcal{P}_{sub}| = 1$. Currently, the occurrence of the matched pair $\{m^a_0, m^b_2\}$ is consistent with $\mathcal{T}$. Therefore, the remaining elements of $\mathcal{T}$ are considered (see line 8 of Algorithm 11.1). Remember, each matched pair specified in the set $\mathcal{P}$ is associated with a set of indices (labelled $\mathcal{E}$) that identify the event numbers of the notes which resulted in the recognition of the pair. Therefore, at this point the current match equals the $|\mathcal{E}|$ associated with $\{m^a_0, m^b_2\}$, which in this case is 4 (see line 17 of Algorithm 11.1). Then, the next pair specified in $\mathcal{T}$ is considered, which is {1, 3} (see line 20 of Algorithm 11.1). From Figure 11.4, it can bee seen that $\{m^a_1, m^b_3\}$ do in fact form a pair which is an element of $\mathcal{P}$. However, the recognition of this pair is inconsistent with the previous pair $\{m^a_0, m^b_2\}$. This is because the pattern specified by $\{m^a_1, m^b_3\}$ does not occur after the pattern specified by $\{m^a_0, m^b_2\}$. Therefore, the note onsets they responded to are not contiguous (see line 24 of Algorithm 11.1). Consequently, this line of enquiry is ended and the current best match is 4.

LOE 2: The next line of enquiry begins with the pair {1, 3}, which is the second phrase specified in $\mathcal{T}$. As the previous line of enquiry has already shown, $\{m^a_1, m^b_3\}$ does form a pair which is an element of $\mathcal{P}$. Therefore, the remaining elements of $\mathcal{T}$ are considered (see line 8 of Algorithm 11.1). At this point the current match equals the length of the pattern encoded by $\{m^a_1, m^b_3\}$, which in this case is 3 (see line 17 of Algorithm 11.1). Then, the next pair specified in $\mathcal{T}$ is considered, which is {2, 4} (see line 20 of Algorithm 11.1). From Figure 11.4, it can bee seen that $\{m^a_2, m^b_4\}$ does in fact form a pair which is an element of $\mathcal{P}$. Furthermore, recognition of this pair is consistent with the previous pair $\{m^a_1, m^b_3\}$. This is because $\{m^a_2, m^b_4\}$ does occur after $\{m^a_1, m^b_3\}$ and the note onsets they are responding to are contiguous (see line 24 of Algorithm 11.1). At this point the match equals 5 ($|\mathcal{E}_{1,3}| + |\mathcal{E}_{2,4}|$). The next pair specified in $\mathcal{T}$ is now considered, which is {3, 5}. Once again, this pair is an element of the set $\mathcal{P}$ and is contiguous with its preceding pair $\{m^a_2, m^b_4\}$. At this point the match equals 9 ($|\mathcal{E}_{1,3}| + |\mathcal{E}_{2,4}| + |\mathcal{E}_{3,5}|$). Then, the next pair specified in $\mathcal{T}$ is considered, which is {4, 6} (see line 20 of Algorithm 11.1). From Figure 11.4, it can bee seen that $\{m^a_4, m^b_6\}$ does in fact form a pair which is an element of $\mathcal{P}$. However, the recognition of this pair is inconsistent with the previous pair $\{m^a_3, m^b_5\}$. This is because $\{m^a_3, m^b_5\}$ is not contiguous with $\{m^a_4, m^b_5\}$. Consequently, this line of enquiry is ended with a match value of 9.

LOE 3 and 4: Since the second, third and fourth pairs in the set $\mathcal{T}$ combined to yield a match, the next lines of enquiry begin with the fifth entry in $\mathcal{T}$, which is $\{4, 6\}$, and the final entry in $\mathcal{T}$, which is $\{5, 9\}$. However, similarly to the first line of enquiry, although both $\{m^a_4, m^b_6\}$ and $\{m^a_5, m^b_9\}$ are elements of $\mathcal{P}$, these pairs are not contiguous and consequently yield match values of 2 and 4 respectively.

From examining these lines of enquiry, it is clear that the set $\mathcal{P}_{seq} = \{\{1,3\},\{2,4\},\{3,5\}\}$ constitutes the best match. Since $|\mathcal{P}_{seq}| = 9$, a match value of 9 is returned for this template node.

11.6 Summary

This chapter has introduced the *Re*TREE*ve* algorithm which can be populated using the boundary points discovered using SONNET-MAP. Furthermore, the information contained in *Re*TREE*ve*'s LTM can be utilized to perform melody retrieval using Algorithm 11.1. Together, the SONNET-MAP and *Re*TREE*ve* networks constitute the first multidimensional, self-organizing neural system to be applied to the problem of CBR of melodies, where a unified framework tackles the issues of segmentation, learning, recognition and retrieval. Chapter 13 will describe the empirical results obtained using *Re*TREE*ve*.

Part IV

Empirical Results

Chapter 12

Segmenting Melodies using SONNET-MAP

12.1 Introduction

Chapters 8, 9 and 10 have shown how parallel SONNET modules can process different facets of a melody simultaneously using the SONNET-MAP architecture. In this chapter, which presents the largest empirical test of a SONNET-based network reported to date, it is proven that:

1. SONNET-MAP can form multidimensional melodic phrases where each dimension of a phrase is represented independently by parallel SONNET modules and unified through the use of associative map fields.
2. SONNET-MAP is capable of discovering plausible boundary points within a melody using a pitch-interval and IOI representation, where grouping cues such as proximity, repetition and STM limitations influence the learning of patterns.
3. Illustrate the benefit of utilizing parallel SONNET modules to segment melodies.

The boundary points discovered during these experiments will be utilized in Chapter 13 to efficiently populate a *Re*TREE*ve* network with the entire collection of the *Beatles* melodies. From this, the *Re*TREE*ve* CBR algorithm (introduced in Chapter 11) will be empirically tested.

12.2 Experimental setup

12.2.1 Using pitch-interval and IOI representations

Section 3.3 discussed possible representations for the pitch and rhythm dimensions of a melody — IPCs, pitch-intervals, melodic contours and IOIs. Additionally, Chapter 9 discussed the manner by which these representations can be encoded within the framework of a SONNET-based neural architecture.

With regard to representing IOIs at $\mathcal{F}_1$, Roberts has convincingly shown that a SONNET module operating in time-driven mode, and using a SSG architecture to represent IOIs, better models aspects of human rhythm perception than one operating in event-driven mode, utilizing an MSG architecture to represent quantized IOIs. Consequently, in line with Roberts, the same IOI representational scheme will also be employed in this chapter.

With regards to representing pitch sequences, either an IPC representation or a pitch-interval representation should be utilized. This is because melodic contours do not provide enough information from which to produce adequate segmentations. The choice made here is based purely on practical grounds. The use of the IPC representation with the IOI representation is simply incompatible. This is due to the following reasons:

1. Suppose that the first five notes from the melody *Day Tripper* have been learned by SONNET-MAP. Furthermore, suppose IPCs are used to represent pitches and IOIs are used to represent rhythms. In this case, pitch information on all five notes will be completely represented by the following IPC sequence {5, 4, 2, 0, 0}. However, rhythm information on the same five notes will be represented by the following IOI sequence {0.5, 0.5, 0.5, 1.0}, because the first interval is only created when the second note occurs, and so on. From this example, it can be seen that there is a mismatch between the amount of information contained within each sequence, although $|T_{ni}| = 5$ in both cases.

2. Due to their respective chunking-out protocols, particularly the K_{chunk} setting, the last onset of an IOI pattern is not chunked-out (for the reason discussed in Section 7.3.4). However, the last onset of an IPC pattern is chunked-out. Consequently, the working memories of the pitch and rhythm modules will become inconsistent after chunking-out occurs.

As a result, a pitch-interval representation is preferred. Furthermore, pitch-intervals should be represented using a SSG architecture with CARs, which is precisely analogous to the IOI representation utilized for rhythms. Additionally, the use of pitch-intervals enables segmentation at the pitch module to be biased towards proximal pitches, which is an additional advantage of using this scheme. This is the first time that a SONNET-based network has been utilized to process pitch-intervals in this manner.

Finally, once boundary points have been discovered, a *Re*TREE*ve* network can be fast-trained with any of the representational schemes presented in Chapter 9. This is particularly useful for CBR because, in certain circumstances, some representations are better for the purposes of retrieval than others (see Chapter 13).

12.2.2 Mode of operation

The mode of operation employed by each SONNET-MAP module has a significant effect on the segmentation process. That is, whether a module is operating in event-driven mode or time-driven mode (see Section 6.3). In the simulations presented in this chapter, the pitch module operates in event-driven mode, while the rhythm module operates in time-driven mode. In event-driven mode, a network adapts its cells' activations and weights for a time period of T_{attn} after an event has occurred. On the other hand, in time-driven mode a network continuously adapts its cells' activations and weights, regardless of whether an event has occurred or not[1]. Additionally, the mode of operation significantly affects when input assemblies at $\mathcal{F}_1$ may be reset, either due to $\mathcal{F}_1$ overload or to the chunking-out process at $\mathcal{F}_2$.

$\mathcal{F}_1$ overload. Together with the mode of operation, the parameters μ, ω and K_{12} are used to specify the behaviour of the input field at $\mathcal{F}_1$. When an event occurs, the appropriate input assembly is set to μ, which is equal to unity for both the pitch and rhythm modules. Thereafter, the input assembly increases at a rate of ω, which is set to 3 for the pitch and rhythm modules in this this chapter. K_{12} specifies the overload threshold, above which input assemblies are reset. The actual cells that are reset depend on the overload scheme employed by the module in question (see Section 5.3.1). For the pitch module $K_{12} = 7$, while for the rhythm module $K_{12} = 6$. In event-driven mode, K_{12} represents the time in terms of T_{attn}. Meanwhile, in time-driven mode, K_{12} represents the time in terms of T_{span}.

Chunking-out at $\mathcal{F}_2$. Together with the mode of operation, the parameters K_{13} (which is generally equal to $B/2$), K_{14} and K_{chunk} govern the chunking-out process. For pitch sequences and rhythms, the last onset of a pattern is not reset during the chunking-out process ($K_{chunk} = false$). In event-driven mode, K_{14} represents the time in terms of T_{attn}. Meanwhile, in time-driven mode, K_{14} represents the time in terms of T_{span}.

An additional point must also be discussed with respect to event-driven networks and the value afforded to T_{attn}. Due to the wide variety of IOIs inherent in melodies, the time period of T_{attn} may not have elapsed by the time the subsequent note has occurred. Thus, on certain occasions the network will not adjust its cells and weights for the entire duration of T_{attn} but instead, for the duration of the IOI. For example, the duration of the shortest IOI in the set of *Beatles* melodies is 95ms (from *All My Loving*). T_{attn} could be set in accordance with the shortest IOI in an input set. However, for real-time systems, this quantity would be impossible to know in advance. Furthermore, if T_{attn} is set too small, extremely slow learning will ensue, which was discovered to be the case when T_{attn} was set to 95ms during preliminary SONNET-MAP simulations. Consequently, T_{attn} should be considered as the maximum time for which an event-driven SONNET module may adjust its cells and weights after a note has occurred.

12.2.3 Architectural configuration

The architectural configuration of the SONNET-MAP network used in this chapter is described in this section (see Table 12.1 and Section A.2).

[1] Remember, cells and weights are calculated at points in time that are separated by the time interval T_{step} (see Section A.4.1).

Parameter		$\mathcal{S}^a$	$\mathcal{S}^b$
$\mathcal{F}_0$	N_0	88	
$\mathcal{F}_1$	N_g	1	1
	N_s	30	37
$\mathcal{F}_2$	N_g	30	30
	N_s	7	7
	N_l	5	5
	K_{zmin}	0.06	0.06
	K_{zmax}	0.08	0.08
	K_{winit}	0.05	0.05
Maps	K_{uinit}	0.1	0.1

Table 12.1: The architectural configuration of the SONNET-MAP network used in this chapter, where $\mathcal{S}^a$ represents the pitch module and $\mathcal{S}^b$ represents the rhythm module.

Event field at $\mathcal{F}_0$. $\mathcal{F}_0$ is divided into a field containing 88 cells, each responding to a particular CPH. $\mathcal{F}_0$ is connected to $\mathcal{F}_1$ using the appropriate gating mechanism and in accordance with the representational scheme employed. See Section 9.2.1 for more detail.

Input fields at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$. $\mathcal{F}_1^a$ (at the pitch module) consists of a single sibling group containing thirty siblings. Although only a maximum of seven items can be held in working memory at any particular point in time, the additional siblings are required during the segmentation phase. This issue is discussed further in Section 12.2.6. $\mathcal{F}_1^b$ (at the rhythm module) consists of a single sibling group containing thirty-seven siblings, which enables melodies containing a rapid succession of notes to be represented adequately.

Masking fields at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$. At $\mathcal{F}_2$, both the pitch module and the rhythm module contain thirty sibling groups, each containing seven siblings. Thirty sibling groups are chosen to ensure that there are a sufficient number of cell assemblies available to represent the patterns learned from each melody. However, this number is kept at a minimum so as to reduce computation time. Seven siblings (N_g^2) are chosen in each group so that boundary point alignment can operate correctly, whilst also modelling the human STM span.

Link architecture. The pitch module and rhythm module contain five interacting links between each $\gamma - \chi$ pair. The initial bottom-up weights on these links should be carefully chosen; that is, the values for K_{zmin} and K_{zmax}. For instance, these should be set such that the initial value of z_{ni}^{tot} does not exceed K_5, because a cell assembly is considered to have a reasonable pattern representation when $z_{ni}^{tot} > K_5$. This would be inconsistent with the fact that no learning has taken place yet. Additionally, in order to prevent extremely slow initial learning, z_{ni}^{tot} should not be too small. In other words, K_{zmin} and K_{zmax} must be set in accordance with the minimum bottom-up excitatory weight vector length, z_{min}. Remember,

z_{min} is used to prevent an excessive slowdown in learning times by setting a floor value, which z_{ni}^{tot} may not fall below. Typically, z_{min} is set between 0.05 and 0.1. In the simulations presented in this chapter, initial values for K_{zmin} and K_{zmax} are set such that z_{ni}^{tot} marginally exceeds z_{min}. These initial parameter values can have a profound effect on the segmentation process. For the pitch module, having K_{zmin} and K_{zmax} set over a narrow range biases cell assemblies towards phrases containing proximal pitches. Likewise, for the rhythm module, having K_{zmin} and K_{zmax} specified over a narrow range biases cell assemblies to perform grouping by temporal proximity.

Map fields. The map fields connecting $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ ($\mathcal{F}_1^{ab}$ and $\mathcal{F}_1^{ba}$), and the map fields connecting $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ ($\mathcal{F}_2^{ab}$ and $\mathcal{F}_2^{ba}$) mirror the architecture of their source fields (see Chapter 10). For example, the map field connecting $\mathcal{F}_2^b$ to $\mathcal{F}_2^a$ ($\mathcal{F}_2^{ba}$) will have an identical configuration to the masking field $\mathcal{F}_2^a$, that is $N_g^{ba} = 30$ and $N_s^{ba} = 7$. Initially, the adjustable weights at the $\mathcal{F}_1$ map fields are set to unity. Meanwhile, the adjustable weights at the $\mathcal{F}_2$ map fields are set to K_{uinit} (which typically equals 0.1). Finally, $K_{althresh}$ (the associative learning threshold) is set to 0.2 ($K_{uinit} + v_c$, where $v_c = 0.1$).

12.2.4 Parameter settings

A variety of parameters are used in the operational specification of the SONNET-MAP network. The use of these parameters has been discussed during the presentation of the networks in Chapters 5 through 11 and indeed, some have already been discussed in this chapter. In many instances, it is adequate to simply examine the function played by a particular parameter when setting its value. In other instances more careful consideration is necessary, where the role played by a particular parameter is best determined under experimental conditions. For other discussions on setting SONNET-based parameters, see Nigrin [114], Page [118, Section 6.5.2] and Roberts [133, Section 3.3.4].

ϵ_1 is the excitatory learning rate. This parameter can be set to enable fast, single-shot learning, where a cell assembly can become committed to a pattern only after a single trial. Slow to moderate learning rates allow SONNET to generalize in a context-sensitive manner over multiple epochs. To discover the sequential regularities embedded in the pitch-interval sequences and the IOI sequences, a moderate learning rate is used.

v_1 and v_2 are constants used to specify different path strengths between cell assemblies at $\mathcal{F}_2$. These parameters allow the dynamics of competition to be altered by varying the ratio between excitatory and inhibitory input signals [114, pg. 119]. v_1 and v_2 also play an important role in the learning rate [133, Section 3.3.4].

$K_\times$ is used to calculate $I_{ni}^{\times}$, which ultimately specifies how a cell assembly's size, D_{ni}, increases as the length of its pattern increases. In the simulations presented in this chapter $K_\times = 1$. Consequently, D_{ni} will grow by a power of two per increment in $|T_{ni}|$. In other words, $D_{ni} = |T_{ni}|^{K_\times + 1}$ for committed cell assemblies.

$\boldsymbol{K_5}$ represents the value z_{ni}^{tot} must reach before a cell assembly is considered to have a reasonable pattern representation. Thus, z_{ni}^{tot} measures the strength with which χ_{ni} represents its pattern. Higher values indicate better pattern representation. When z_{ni}^{tot} exceeds the K_5 threshold, z_{ni}^{tot} is considered to have performed a significant amount of learning. At this point the learning rate modulator is utilized, which increases the stability of χ_{ni}'s pattern representation. Typically, K_5 is set to between 0.25 and 0.3. Additionally, K_{embed} (used for repetition handling) should take into account the value of K_5. Since repetition handling is only allowed to occur once a cell assembly has a good pattern representation, K_{embed} should be greater than K_5.

$\boldsymbol{K_3}$ is used to determine whether a secondary weight is sufficiently large to become a member of the set T_{ni}. Thus, K_3 has an influence on the length of encoded patterns. The value of K_3, which should always be set greater than 1, has consequences for the number of contexts required to learn a pattern that is embedded within a larger pattern. In other words, the ease at which generalization can occur. Relatively low values for K_3 reduce the number of contexts required for generalization, while relatively high values for K_3 increases the number of contexts required. Consequently, low K_3 values make it easier for smaller patterns to be learned. Therefore, K_3 should be optimized and chosen carefully according to the particular types of input sequences to be learned. The learning rate should also play a role in selecting an appropriate value for K_3. When fast learning is required, the value of K_3 is not particularly significant. However, when attempting to learn temporal regularities from multiple trials, the learning rate will usually be relatively small and K_3 should be chosen carefully.

$\boldsymbol{K_7}$ contributes to the ceiling placed on the maximum inhibition an uncommitted cell assembly may receive and is instrumental in determining the speed at which patterns that are similar to previously learned patterns can be learned. Relatively low values aid learning when the sequences contain many similar event types, although extremely low values may cause unstable learning and obstruct the chunking-out of previously encoded patterns. Since melodies contain many phrases with similar notes, K_7 was set relatively small.

$\boldsymbol{q}$ specifies the minimum number of cell assemblies that should respond to a given pattern across $\mathcal{F}_1$. Since run-times are approximately proportional to $(N_2^g \times N_2^s)^2$, it is desirable to keep q fairly small in order to minimize computation times.

$\boldsymbol{K_I}$ Setting K_I for pitch-interval or IOI representations requires taking the worst case scenario in each case into consideration. For pitch-intervals, a pattern consisting of four identical pitch-intervals derived from a phrase consisting of five identical pitches represents the worst case scenario. This is because, when the first two onsets of such a sequence is presented, the difference between Z_{jni} and S_{jni} is maximal. In this situation, K_I should be set to 0.75 (see Table 12.2). Similarly for IOI representations, the worst case scenario is generally a pattern consisting of four IOIs, each equal to one quarter of a tactus-span. In this situation K_I should be set to to be set to 0.84 (see Table 12.3).

Link index (j)	**1**	**2**	**3**	**4**	**5**
Z_{jni}	0.567	0.494	0.431	0.376	0.327
S_{jni}	0.754	0.657	0	0	0
$I^{\times}_{jni}$	0.752	0.752	0	0	0

Table 12.2: Setting K_I for the pitch-interval representation. With $\omega = 3$, $K_{intervals} = 8$ and $N_l = 5$, K_I should be set to 0.75. This is derived from considering the $I^{\times}_{jni}$ value in the worst case scenario, where a pattern consisting of four identical pitch-intervals is derived from a phrase consisting of five identical pitches (represented by five event onsets at $\mathcal{F}_1$).

Link index (j)	**1**	**2**	**3**	**4**	**5**
Z_{jni}	0.672	0.511	0.388	0.295	0.224
S_{jni}	0.796	0.605	0	0	0
$I^{\times}_{jni}$	0.844	0.844	0	0	0

Table 12.3: Setting K_I for the IOI representation. With $\omega = 3$, and $N_l = 5$, K_I should be set to 0.84. This is derived from considering the the $I^{\times}_{jni}$ value in the general worst case scenario, where a pattern consists of four IOIs each equal to one quarter of a tactus-span (represented by five event onsets at $\mathcal{F}_1$).

12.2.5 Presentation regime

Each melody is presented to SONNET-MAP repeatedly until the prescribed number of epochs (K_{epoch}) has elapsed. After each epoch, the entire SONNET-MAP network (its cell activations) is reset in preparation for the forthcoming epoch. After each melody cycle, the network's LTM state and simulation logs are saved (see Sections D.6 and D.7). Then, a new network is created to process the subsequent melody. Thus, only a single SONNET-MAP network is utilized per melody. This enables the network to remain small and keep run times at a minimum. Indeed, a similar learning strategy was employed by Roberts [133, Chapters 5, 6 and 7].

The number of epochs employed for each melody should generally be determined by the size of the learning rate (ϵ_1). Low leaning rates require more epochs, while high learning rates require fewer epochs. In the simulations presented in this chapter, for each melody the pitch module runs for 30 epochs, while the rhythm module runs for 10 epochs. As a consequence, the number of epochs required by the pitch module differs from that of the rhythm module. Therefore, after the 10^{th} epoch has expired, the rhythm module is disabled for the remainder of the pitch module's epochs.

Additionally, an isochronous sequence of 7 tactus-span beats is appended to each melody. This has the effect of giving the final phrase of each melody an additional time period to learn, which helps model the recency effect.

Parameters	**Pitch module**	**Rhythm module**
A	1	1
B	2	2
$K_{althresh}$	0.2	0.2
K_{chunk}	false	false
K_D	0.01	0.01
$K_{encoding}$	pitch-interval using CARs	IOI using CARs
K_{epoch}	30	10
K_{embed}	0.4	0.4
K_I	0.75	0.84
K_{inh}	2000	2000
K_{latch}	0.004	0.004
K_M	0.95	0.95
K_{mode}	event-driven	time-driven
$K_\times$	1	1
K_3	1.5	2
K_5	0.25	0.25
K_7	50	50
K_{10}	0.5	0.5
K_{12}	7 events	6 tactus-spans (T_{span})
K_{13}	1	1
K_{14}	1 attentional pulse (T_{attn})	1 tactus-span (T_{span})
q	2	2
T_{attn}	150ms	N/A
T_{span}	N/A	dependent on melody
T_{step}	2ms	2ms
v_1	3	2
v_2	30	30
z_{min}	0.1	0.1
ϵ_1	4	2
ϵ_2	10	20
ϵ_3	0.01	0.01
μ	1	1
ρ	0.95	0.95
ω	3	3

Table 12.4: Parameter settings (alphabetically ordered).

12.2.6 Performing the segmentation

A melody segmentation is formed when the patterns learned by SONNET-MAP are chunked-out. The boundary points bracing each chunked-out pattern are directly recorded in the input files which represent each melody, where the beginning and ending of each pattern is clearly distinguished. However, during extensive preliminary SONNET-MAP simulations, a number of issues were discovered which sometimes obstructed the desired chunking-out of the patterns learned by SONNET-MAP. This hampered the formation of consistent and clean segmentations during the prescribed number of epochs. These problems stem from the fact that SONNET-MAP's operation was optimized to discover regularities within input sequences. Consequently, the network is continually learning and revising the manner in which input sequences are processed. However, when the goal is simply to extract the information contained within SONNET-MAP at a particular point in time (in this case, after a fixed number of epochs have elapsed), SONNET-MAP's operation must be adapted to achieve this. Consequently, the actual segmentations which are presented in Section 12.4 are generated during an additional step called the *segmentation phase.* This step is simply an additional epoch dedicated exclusively to generating a segmentation based on the patterns discovered by SONNET-MAP during normal network operation. The following operational modifications are introduced during the segmentation phase:

$\mathcal{F}_1$ overload disabled. In some circumstances, there may be a gap of several notes between two patterns learned in a melody. If this gap is sufficiently large, an $\mathcal{F}_1$ overload will occur. This overload may cause part of the subsequent pattern to be deleted from $\mathcal{F}_1$. Thus, the second pattern may not be recognized at all and may never form part of the segmentation. Obviously, this is undesirable. For example, suppose the pitch-interval patterns {1, 1, 2} and {3, 3, 3, 3} have been learned. Furthermore, suppose these patterns occur within the following sequence {0, *1*, *1*, *2*, 4, 4, *3*, *3*, *3*, *3*, 1}. In this circumstance, the pattern {1, 1, 2} will be chunked-out soon after it has occurred. Now, just after the onset which defines the last pitch-interval of the second pattern has occurred, the $\mathcal{F}_1$ overload threshold will have been exceeded (remember that $K_{12} = 7$ events). Depending on the random element, the first item of the second pattern may be completely or partially deleted from $\mathcal{F}_1$. If this transpires, the second pattern will no longer be completely present at $\mathcal{F}_1$ and consequently will not be recognized nor chunked-out. To overcome this problem, the $\mathcal{F}_1$ overload mechanism is disabled during the segmentation phase.

Chunking-out is extended. As a consequence of disabling $\mathcal{F}_1$ overloads, some events at $\mathcal{F}_1$ may not be reset at all. These include events that do not form part of any pattern. To overcome this, when chunking-out, events older than the oldest event being chunked-out are also deleted from $\mathcal{F}_1$.

Restricted cell assembly activation. To avoid interference from uncommitted cell assemblies whose partially encoded patterns may overlap with committed cell assemblies, only committed cell assemblies may activate during the segmentation phase. This is similar to Pages' expectancy work and use of the set P. This prevents competition from uncommitted cell assemblies from interfering with the chunking-out process of committed cell assemblies. Furthermore, only cell assemblies that represent patterns greater than two intervals may activate during the segmentation phase (that is, $|T_{ni}| \geq 3$). This restriction is purely intended to aid in the CBR experiments presented in Chapter 13, because larger patterns provide greater discriminatory power for retrieval purposes

Restricted module interaction. Modules may only interact during the segmentation phase. This allows pitch-interval sequences and IOI sequences to form independently from one another. This restriction enables the nuances of each representation to contribute to the segmentation process. See Section 10.4.1 for more details on this approach.

Rhythm module dominates pitch module. One intricacy related to the interaction of parallel modules during the segmentation phase regards the actual chunking-out of learned phrases. Unless this is controlled in the following way, the final segmentation formed may become a mishmash between patterns formed at the rhythm module and the pitch module. To avoid this potential problem, one module will dominate the segmentation process during this phase. Since, as was discussed in Chapter 2, time dominates the perception of melodies, the IOI module will dominate the pitch-interval module. This is achieved by suppressing the pitch module when any cell assembly at the rhythm module recognizes its full pattern; that is, $\exists\ \chi_{ni} \mid M_{ni} \geq K_M$. In this circumstance, the attentional signal at the pitch module will be set to, and remain set to, zero until such a time as $\neg\exists\ \chi_{ni} \mid M_{ni} \geq K_M$ at the rhythm module. This approach has the effect of segmenting each melody based of the rhythmic content of the melody, with certain segments not classified by the rhythm module being classified by the pitch module.

Although SONNET-MAP frequently learns every note of a melody, it is possible that some notes remain unlearned during normal network operation. This could have been solved by simply increasing the learning rate, ϵ_1. However, as we have already mentioned, SONNET-MAP's parameters were optimized to discover regularities within input sequences. Consequently, the learning rate was not sufficiently high to learn every note during the specified number of epochs. Another solution would have been to increase the number of epochs used. However, this would have undesirably increased network run-times. Since it is desirable for the *Re*TREE*ve* network to learn every note of every melody in the *Beatles* test suite, we introduce a simple algorithm to group the remaining unlearned notes into phrases. This algorithm is solely used for efficiency reasons. Therefore, after the segmentation phase, the input files used to record each melody segmentation are further processed to eliminate any *gaps* which may exist. This step is required so that the *Re*TREE*ve* network can be populated with complete melodies.

Generally, when a sequence of notes have not been learned the resulting group usually consists of N_l onsets or less. In this circumstance, since the maximum group size allowed is N_l (remember $N_l = 5$), these onsets are simply bundled into a single phrase. In circumstances where this sequence of notes is greater than N_l, the sequence is processed in chunks of N_l in the following way. The the largest IOI within a chunk will indicate a phrase boundary. In cases where there is a tie, the IOI closest to the end of the melody is chosen, which guarantees that the patterns formed are as large as possible. Following that, the next N_l note onsets are chosen from the point at which the previously grouped pattern ends. The process continues in this manner until the entire sequence of unlearned notes have been grouped. Similar to the chunking-out process, the boundary points bracing each grouped pattern are directly recorded in the input files which represent each melody.

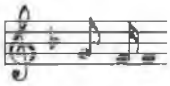
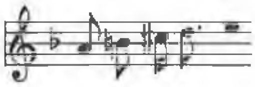
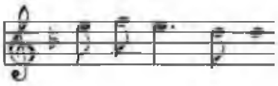
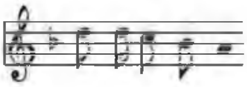
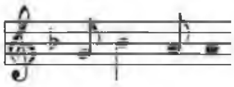
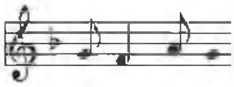
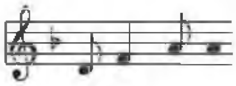
12.3 Interpreting the results

The segmentation results for ten of the melodies used in this thesis are presented in Section 12.4. The segmentation for each melody is described using three tables. The first table lists each pattern, which contributes to the segmentation of the melody, learned by SONNET-MAP. The order in which each pattern was committed is also reflected in this table. Each pattern is labelled and illustrated in music notation. Furthermore, the primary factors that influenced the formation of each pattern are also listed (see Section 2.7 for more information on these factors). Each factor is abbreviated as shown in Table 12.5. Note also, due to the fact that interval representations are being used, the last note-head of each phrase is simply used to complete the interval of each phrase. The second table illustrates the order that each pattern illustrated in the first table is chunked-out. Replacing the labels associated with each pattern will yield the entire melody segmentation in music notation. This table also reflects any higher level structure that can be inferred from the segmentation performed. For example, verse and chorus repetitions or the beginning of a new section. The final table illustrates the placement of boundary points using pitch-interval and IOI notation. Each segment is accompanied by its music notation label and melody lyrics. This table is intended to give the reader a greater feel and understanding of each of the formed segmentations. Note also, the final note of some phrases are accompanied by the ‡ symbol. This means that the interval information at this point has not been learned, only the fact that a note onset occurs. In almost every case, this occurs at places where relatively large IOIs exist and SONNET-MAP was unable to learn the IOI. This is consistent with studies in human rhythm perception which has found that humans find long IOIs difficult to learn.

alg	Segments formed due to unlearned notes during normal SONNET-MAP operation.
chunk	Influence from previously chunked-out patterns.
ctx	Multiple contexts.
dom	The dimension which influenced the formation of the pattern. That is, either pitch or rhythm; *dom(p)* or *dom(r)* respectively.
gap	Gap principle.
prim	Primacy effect.
prox	Pitch proximity, *prox(p)*, or temporal proximity, *prox(t)*
rec	Recency effect.
rep	Pitch repetition, *rep(p)*, or rhythm repetition, *rep(r)*.

Table 12.5: Factors influencing segmentation. Note that, when the dominance factor is present, all other grouping factors relate to the dimension which is dominant. For example, if *dom(r)* and *prox* are listed, the proximity factor relates to temporal proximity. Where no dominance factor is specified, *prox* would relate to both temporal proximity and pitch proximity unless explicitly stated otherwise.

12.4 Segmentations

Label	Phrase	Factors
$\mathcal{A}$		*dom(r), gap, prim*
$\mathcal{B}$		*prox, rep(p)*
$\mathcal{C}$		*chunk, gap, prox(p)*
$\mathcal{D}$		*dom(r), prox*
$\mathcal{E}$		*ctx, dom(r), gap*
$\mathcal{F}$		*ctx, dom(r), gap*
$\mathcal{G}$		*ctx, dom(r), gap*

(a)

Figure 12.1: The boundary points discovered for the melody *Yesterday* (see Appendix E.4.1). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

$\mathcal{A}$ $\mathcal{B}$ $\mathcal{C}$

$\mathcal{D}$ $\mathcal{E}$

$\mathcal{F}$ $\mathcal{G}$

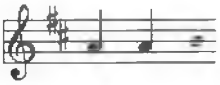
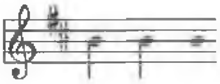
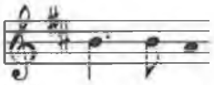
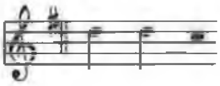
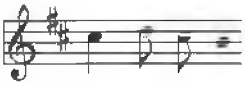
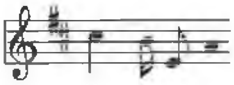
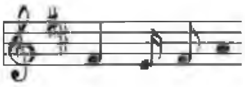
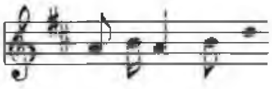
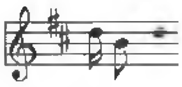
(b)

	$\mathcal{A}$			$\mathcal{B}$					$\mathcal{C}$		
2	0	4‡	2	2	1	2	1	1	2	0	0‡
0.5	0.25	4.25‡	0.5	0.5	0.25	0.75	0.5	0.5	1.5	0.5	3.0‡
Yes-	-ter-	-day	all	my	trou-	-bles	seemed	so	far	a-	-way

		$\mathcal{D}$				$\mathcal{E}$		
0	2	2	1	2‡	3	1	0	2‡
0.5	0.25	0.5	0.5	0.75‡	0.5	1.0	0.5	2.0‡
Now	it	looks	as	though	they're	here	to	stay

	$\mathcal{F}$				$\mathcal{G}$		
2	4	2	5‡	3	4	0	‡
0.5	1.0	0.5	2.0‡	0.5	1.0	0.5	‡
Oh	I	be-	-lieve	in	yes-	-ter-	-day

(c)

Label	Phrase	Factors
$\mathcal{A}$		*dom(r), prim, rep*
$\mathcal{B}$		*dom(r), rep*
$\mathcal{C}$		*chunk, dom(r), gap*
$\mathcal{D}$		*dom(r), rep*
$\mathcal{E}$		*chunk, dom(r), gap*
$\mathcal{F}$		*ctx, dom(r), gap*
$\mathcal{G}$		*ctx, dom(r), gap*
$\mathcal{H}$		*chunk, dom(r)*
$\mathcal{I}$		*chunk, ctx, dom(r), prox, rec*

(a)

Figure 12.2: The boundary points discovered for the melody *Mother Nature's Son* (see Appendix E.4.2). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

$\mathcal{A}$ $\mathcal{B}$ $\mathcal{C}$

$\mathcal{D}$ $\mathcal{E}$

$\mathcal{F}$ $\mathcal{G}$ $\mathcal{H}$ $\mathcal{I}$

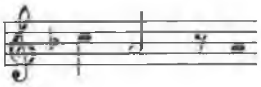
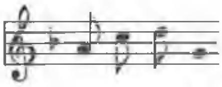
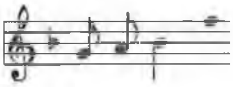
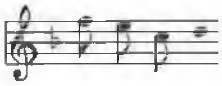
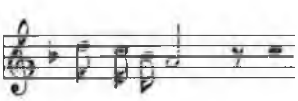
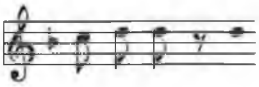
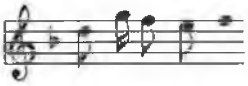
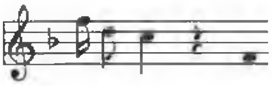
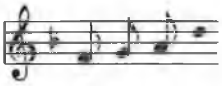
(b)

$\mathcal{A}$		$\mathcal{B}$		$\mathcal{C}$			$\mathcal{D}$	
0	2	0	0	0	2	5‡	0	1
1.0	1.0	1.0	1.0	1.5	0.5	2.0‡	1.0	1.0
Born	a	poor	young	coun-	-try	boy	Mo-	-ther

$\mathcal{E}$				$\mathcal{F}$			
1	1	2	2‡	2	7	5	3‡
1.0	0.5	0.5	4.0‡	1.0	0.25	0.5	1.25‡
na-	-tu-	-re's	Son	All	day	—	long

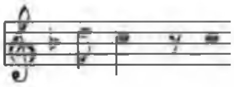
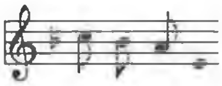
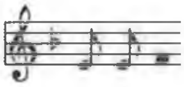
$\mathcal{G}$			$\mathcal{H}$				$\mathcal{I}$		
2	2	3	2	2	2	3	3	3	‡
1.0	0.25	0.5	0.5	0.25	1.0	0.5	0.25	0.5	‡
I'm	sit-	-ing	sing-	-ing	songs	for	eve-	-ry	one

(c)

Label	Phrase	Factors
$\mathcal{A}$		*alg*
$\mathcal{B}$		*ctx, dom(r), prox*
$\mathcal{C}$		*ctx, dom(r)*
$\mathcal{D}$		*ctx, dom(r), prox*
$\mathcal{E}$		*chunk, dom(r), gap, prox*
$\mathcal{F}$		*ctx, dom(r), prox*
$\mathcal{G}$		*chunk, dom(r), prox*
$\mathcal{H}$		*dom(p), ctx*
$\mathcal{I}$		*ctx, dom(r), prox*

(a)

Figure 12.3: The boundary points discovered for the melody *Hey Jude* (see Appendix E.4.3). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

Label	Phrase	Factors
$\mathcal{J}$		*alg*
$\mathcal{K}$		*ctx, dom(r), prox*
$\mathcal{L}$		*alg*

(a) (continued)

$\mathcal{A}$ $\mathcal{B}$ $\mathcal{C}$ $\mathcal{D}$ $\mathcal{E}$

$\mathcal{F}$ $\mathcal{G}$ $\mathcal{H}$

$\mathcal{I}$ $\mathcal{J}$ $\mathcal{K}$ $\mathcal{L}$

(b)

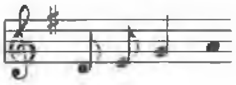
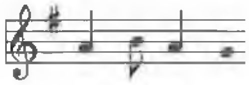
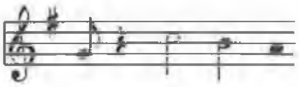
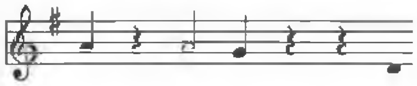
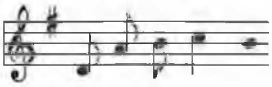
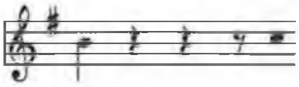
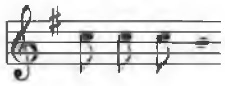
$\mathcal{A}$		$\mathcal{B}$				$\mathcal{C}$			
3	0	3	2	7	0‡	2	1	7	0‡
1.0	2.5	0.5	0.5	0.5	3.0‡	0.5	0.5	1.0	1.5‡
Hey	Jude	don't	make	it	bad	take	a	sad	song

$\mathcal{D}$			$\mathcal{E}$				$\mathcal{F}$		
1	4	2	2	2	1	3	2	0	0
0.5	0.5	0.5	0.5	0.25	0.25	2.5	0.5	0.5	1.0
and	make	it	bet-	-ter	—	—	Re-	-mem-	-ber

$\mathcal{G}$				$\mathcal{H}$			$\mathcal{I}$		
5	2	1	1	3	2	7	2	2	5
0.5	0.25	0.5	0.5	0.25	0.5	2.0	0.5	0.5	0.5
to	let	her	in-	-to	your	heart	then	you	can

$\mathcal{J}$		$\mathcal{K}$			$\mathcal{L}$		
2	0	2	1	5	0	1	‡
0.5	1.5	0.5	0.5	0.5	0.5	0.5	‡
start	—	to	make	it	—	bet-	-ter

(c)

Label	Phrase	Factors
$\mathcal{A}$		*prim, rep, dom(r)*
$\mathcal{B}$		*gap, prox(p), rep*
$\mathcal{C}$		*chunk, dom(p), ctx*
$\mathcal{D}$		*chunk, dom(p)*
$\mathcal{E}$		*dom(r), gap, prox, rep*
$\mathcal{F}$		*alg*
$\mathcal{G}$		*alg*

(a)

Figure 12.4: The boundary points discovered for the melody *Two of Us* (see Appendix E.4.4). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{B}$	$\mathcal{C}$	
$\mathcal{A}$	$\mathcal{B}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$
$\mathcal{E}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$	

(b)

	$\mathcal{A}$			$\mathcal{B}$				$\mathcal{B}$	
2	3	2	2	2	2	2‡	2	2	2
0.5	0.5	1.0	1.0	0.5	1.0	1.5‡	1.0	0.5	1.0
Two	of	us	rid-	-ing	no-	-where	spend-	-ing	some-

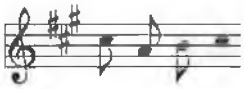
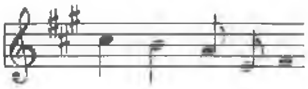
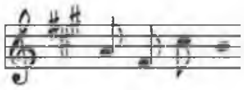
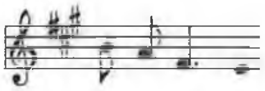
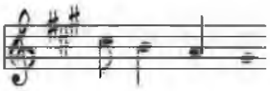
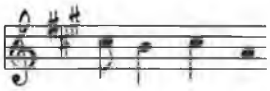
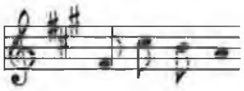
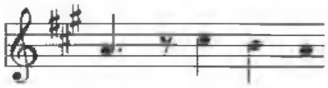
	$\mathcal{C}$				$\mathcal{A}$			$\mathcal{B}$		
5	1	2	7‡	2	3	2	2	2	2	2‡
1.5	2.0	2.0	2.0‡	0.5	0.5	1.0	1.0	0.5	1.0	1.5‡
-one's	hard	earned	pay	You	and	me	Sun-	-day	driv-	-ing

	$\mathcal{B}$			$\mathcal{C}$			$\mathcal{D}$	
2	2	2	5	1	2	0	2	5
1.0	0.5	1.0	1.5	2.0	2.0	2.0	2.0	3.5
not	ar-	-riv-	-ing	on	our	way	back	home

		$\mathcal{E}$					$\mathcal{E}$		$\mathcal{F}$
7	2	1	1	9‡	7	2	1	1	1
0.5	0.5	0.5	1.0	3.5‡	0.5	0.5	0.5	1.0	3.5
we're	on	our	way	home	we're	on	our	way	home

	$\mathcal{G}$		
0	0	1	‡
0.5	0.5	0.5	‡
we're	go-	-ing	home

(c)

Label	Phrase	Factors
$\mathcal{A}$		*gap*, *prim*, *prox*, *rep*
$\mathcal{B}$		*chunk*, *dom(r)*, *rep*
$\mathcal{C}$		*prox(t)*, *rep*
$\mathcal{D}$		*chunk*, *ctx*, *dom(r)*, *gap*
$\mathcal{E}$		*dom(p)*, *prox*
$\mathcal{F}$		*dom(p)*, *prox*, *rep*
$\mathcal{G}$		*prox(t)*, *rep*
$\mathcal{H}$		*alg*

(a)

Figure 12.5: The boundary points discovered for the melody *Here Comes the Sun* (see Appendix E.4.5). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	
$\mathcal{A}$	$\mathcal{B}$	$\mathcal{E}$		
$\mathcal{F}$	$\mathcal{A}$	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{H}$

(b)

$\mathcal{A}$				$\mathcal{B}$				
4	2	2	$0^\ddagger$	2	2	5	2	$3^\ddagger$
0.5	0.5	0.5	$2.0^\ddagger$	1.0	1.0	0.5	0.5	$1.0^\ddagger$
Lit	-tle	dar	-ling	It's	been	a	—	long

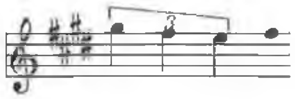
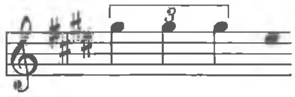
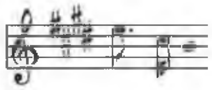
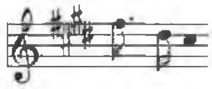
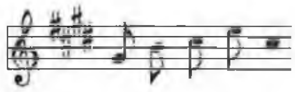
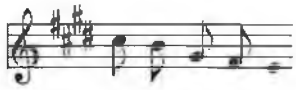
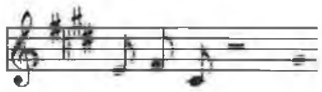
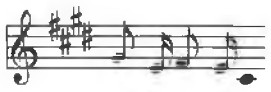
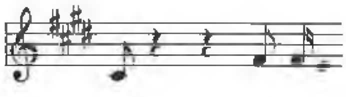
$\mathcal{C}$			$\mathcal{D}$				$\mathcal{A}$			
3	7	2	2	3	2	$9^\ddagger$	4	2	2	$0^\ddagger$
0.5	0.5	0.5	0.5	0.5	1.5	$4.5^\ddagger$	0.5	0.5	0.5	$2.0^\ddagger$
cold	—	lone-	—	-ly	win-	ter	Lit-	-tle	dar-	-ling

$\mathcal{B}$					$\mathcal{E}$			
2	2	5	2	$7^\ddagger$	2	2	1	$5^\ddagger$
1.0	1.0	0.5	0.5	$1.5^\ddagger$	0.5	1.0	1.0	$5.5^\ddagger$
It	feels	like	—	years	since	it's	been	here

$\mathcal{F}$				$\mathcal{A}$			
2	2	4	$4^\ddagger$	4	2	2	$0^\ddagger$
0.5	1.0	1.0	$1.5^\ddagger$	0.5	0.5	0.5	$2.5^\ddagger$
Here	comes	the	sun	do	do	do	do

$\mathcal{F}$				$\mathcal{G}$			$\mathcal{H}$			
2	2	4	$3^\ddagger$	7	2	2	4	2	2	‡
0.5	1.0	1.0	$1.5^\ddagger$	0.5	0.5	0.5	2.0	1.0	1.0	‡
Here	comes	the	sun	and	I	—	say	It's	all	right

(c)

Label	Phrase	Factors
$\mathcal{A}$		*gap, prox, rep*
$\mathcal{B}$		*gap, prox, rep*
$\mathcal{C}$		*chunk, ctx, dom(r), gap*
$\mathcal{D}$		*chunk, ctx, dom(r), gap*
$\mathcal{E}$		*prox, rep*
$\mathcal{F}$		*dom(r), prox, rep*
$\mathcal{G}$		*dom(r), gap, rep*
$\mathcal{H}$		*prox, dom(r), gap*
$\mathcal{I}$		*alg*

(a)

Figure 12.6: The boundary points discovered for the melody *Don't Let Me Down* (see Appendix E.4.6). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$		
$\mathcal{A}$	$\mathcal{D}$	$\mathcal{B}$	$\mathcal{C}$	
$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{H}$	$\mathcal{I}$

(b)

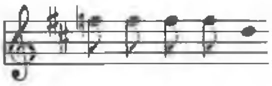
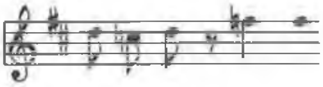
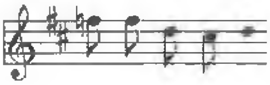
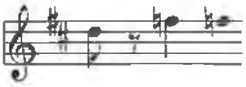
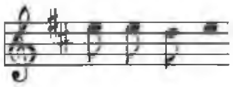
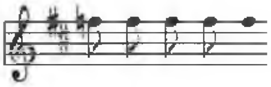
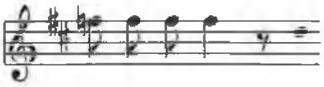
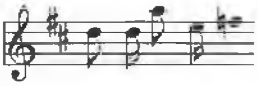
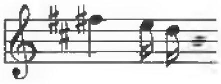
$\mathcal{A}$				$\mathcal{B}$			$\mathcal{C}$		
2	2	2	2‡	0	0	4	3	2	9‡
0.667	0.667	0.667	5.5‡	0.667	0.667	0.667	0.75	0.25	5.0‡
Don't	let	me	down	Don't	let	me	down	—	—

$\mathcal{A}$			$\mathcal{D}$			$\mathcal{B}$			$\mathcal{C}$		
2	2	2	2	3	7‡	0	0	4	3	2	3‡
0.667	0.667	0.667	0.75	0.25	5.0‡	0.667	0.667	0.667	0.75	0.25	3.0‡
Don't	let	me	down	—	—	Don't	let	me	down	—	—

$\mathcal{E}$				$\mathcal{F}$				$\mathcal{G}$		
3	2	3	3	2	3	2	2	2	5	7‡
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	2.5‡
No-	-bo-	-dy	ev-	-er	loved	me	like	she	does	—

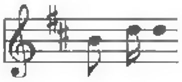
$\mathcal{H}$				$\mathcal{I}$			
4	2	2	3	5	0	2	‡
0.5	0.25	0.5	0.25	2.5	0.5	0.25	‡
Oo	she	does	—	—	yes	she	does

(c)

Label	Phrase	Factors
$\mathcal{A}$		*prox*, *rep*
$\mathcal{B}$		*chunk*, *prox(p)*, *rep*
$\mathcal{C}$		*chunk(p)*, *prox*, *rep*
$\mathcal{D}$		*alg*
$\mathcal{E}$		*ctx*, *dom(r)*, *gap*, *prox*
$\mathcal{F}$		*prox*, *rep*
$\mathcal{G}$		*ctx(r)*, *gap*, *prox*, *rep(p)*
$\mathcal{H}$		*dom(r)*, *gap*, *prox*
$\mathcal{I}$		*chunk*, *dom(r)*, *gap*

(a)

Figure 12.7: The boundary points discovered for the melody *Come Together* (see Appendix E.4.7). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

Label	Phrase	Factors
$\mathcal{J}$		*chunk, dom(r), prox, rec*

(a) (continued)

$\mathcal{A}$ $\mathcal{B}$ $\mathcal{C}$ $\mathcal{D}$ $\mathcal{C}$ $\mathcal{D}$ $\mathcal{C}$

$\mathcal{E}$ $\mathcal{E}$

$\mathcal{F}$ $\mathcal{F}$ $\mathcal{G}$

$\mathcal{H}$ $\mathcal{I}$ $\mathcal{J}$

(b)

𝒜				ℬ			
0	0	0	3	2	2	3	0
0.5	0.5	0.5	0.5	0.5	0.5	1.0	1.0
He	wear	no	shoe	shine	He	got	toe

𝒞				𝒟		𝒞			
0	3	2	2	3	0	0	3	2	2
0.5	0.5	0.5	0.5	1.0	1.0	0.5	0.5	0.5	0.5
jam	foot-	-ball	He	got	mon-	-key	fin-	-ger	He

𝒟		𝒞					ℰ			
3	0	0	3	2	2	2‡	0	2	2	0‡
1.0	1.0	0.5	0.5	0.5	0.5	1.0‡	0.5	0.5	0.5	2.0‡
shoot	Co-	-ca	Co-	-la	He	say	I	know	—	you

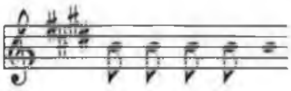
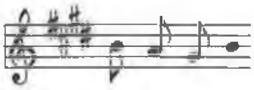
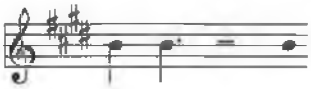
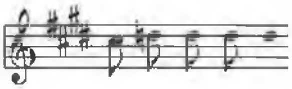
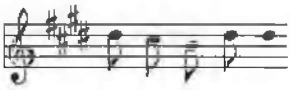
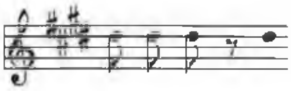
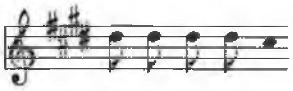
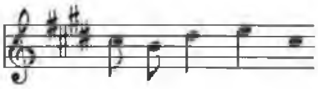
ℰ				ℱ			
0	2	2	1‡	0	0	0	0
0.5	0.5	0.5	2.5‡	0.5	0.5	0.5	0.5
you	know	—	me	One	thing	I	can

ℱ				𝒢			
0	0	0	0	0	0	0	3‡
0.5	0.5	0.5	0.5	0.5	0.5	0.5	1.5‡
tell	you	is	you	got	to	be	free

ℋ					ℐ			
0	7	5	1	1‡	2	2	3	0‡
0.5	0.25	0.5	0.25	2.5‡	1.0	0.25	0.25	2.5‡
Come	to-	-geth-	—	-er	Right	now	—	—

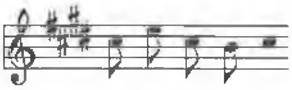
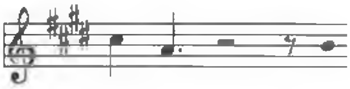
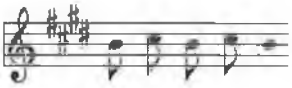
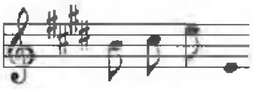
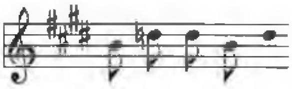
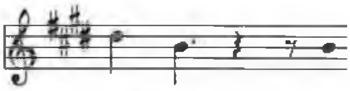
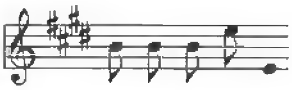
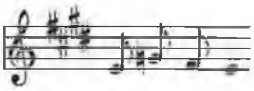
𝒥		
3	0	‡
0.5	0.25	‡
O-	-ver	me

(c)

Label	Phrase	Factors
$\mathcal{A}$		*prox, rep*
$\mathcal{B}$		*ctx, dom(r), gap, prox*
$\mathcal{C}$		*alg*
$\mathcal{D}$		*dom(r), prox, rep*
$\mathcal{E}$		*prox, rep*
$\mathcal{F}$		*alg*
$\mathcal{G}$		*chunk(p), prox, rep*
$\mathcal{H}$		*chunk, dom(r)*

(a)

Figure 12.8: The boundary points discovered for the melody *The Ballad of John and Yoko* (see Appendix E.4.8). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

Label	Phrase	Factors
$\mathcal{I}$		*dom(r), prox, rep*
$\mathcal{J}$		*alg*
$\mathcal{K}$		*dom(r), prox, rep*
$\mathcal{L}$		*dom(r), ctx, gap, prox*
$\mathcal{M}$		*prox, rep(r)*
$\mathcal{N}$		*alg*
$\mathcal{O}$		*dom(r), prox, rep*
$\mathcal{P}$		*dom(r), ctx, gap, prox*

(a) (continued)

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{B}$	
$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{E}$	$\mathcal{G}$	$\mathcal{H}$
$\mathcal{I}$	$\mathcal{J}$	$\mathcal{K}$	$\mathcal{L}$		
$\mathcal{M}$	$\mathcal{N}$	$\mathcal{O}$	$\mathcal{P}$		

(b)

A				B			C	
0	0	0	0	2	1	3	0	0
0.5	0.5	0.5	0.5	0.5	0.5	0.5	1.0	3.5
Stand-	-ing	in	the	dock	in	South-	-amp-	ton

A				B			
0	0	0	0	2	1	3	2‡
0.5	0.5	0.5	0.5	0.5	0.5	0.5	4.0‡
trying	to	get	to	Hol-	-land	or	France

D					E			
1	0	0	0	0‡	1	2	3	0
0.5	0.5	0.5	0.5	1.0‡	0.5	0.5	0.5	0.5
The	man	in	the	mac	said	—	you've	got

F			E				G			
0	0	0	1	2	3	0	0	0	0	1
0.5	0.5	1.0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
to	go	back	you	know	they	did-	-n't	ev-	-en	give

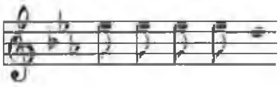
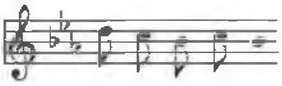
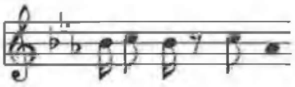
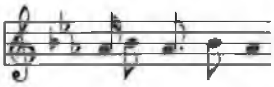
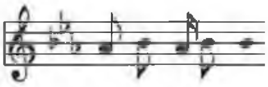
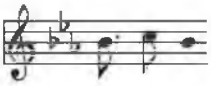
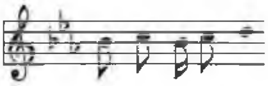
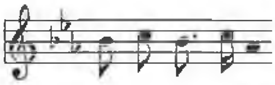
H				I				J	
2	3	2	3	3	3	2	2	4	2
0.5	0.5	1.0	1.0	0.5	0.5	0.5	0.5	1.0	4.0
us	a	chance	Christ!	You	know	it	ain't	ea-	-sy

K				L			
2	2	2	2	2	3	12	7‡
0.5	0.5	0.5	0.5	0.5	0.5	0.5	5.5‡
you	know	how	hard	it	can	be	—

M				N		O			
3	0	3	3	3	0	0	0	5	12
0.5	0.5	0.5	0.5	2.0	3.0	0.5	0.5	0.5	0.5
The	way	things	are	go-	-ing	they're	going	to	cru-

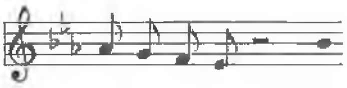
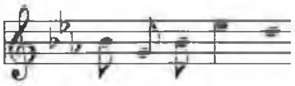
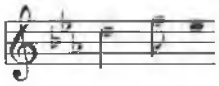
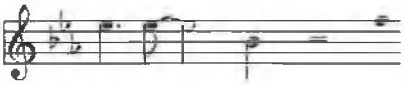
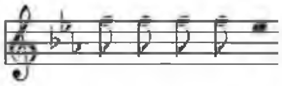
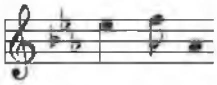
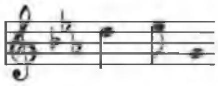
P			
3	1	2	‡
0.5	0.5	0.5	‡
-ci-	-fy	—	me

(c)

Label	Phrase	Factors
$\mathcal{A}$		*prox*, *rep*
$\mathcal{B}$		*prox*, *rep*
$\mathcal{C}$		*chunk*, *prox(p)*, *rep*
$\mathcal{D}$		*chunk*, *prox(p)*, *rep*
$\mathcal{E}$		*dom(r)*, *gap*, *prox*, *rep*
$\mathcal{F}$		*alg*
$\mathcal{G}$		*gap*, *prox*, *rep*
$\mathcal{H}$		*chunk*, *dom(p)*, *prox*, *rep*

(a)

Figure 12.9: The boundary points discovered for the melody *Across the Universe* (see Appendix E.4.9). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

Label	Phrase	Factors
$\mathcal{I}$		*chunk, dom(p)*
$\mathcal{J}$		*alg*
$\mathcal{K}$		*ctx, dom(r), gap*
$\mathcal{L}$		*alg*
$\mathcal{M}$		*dom(r), prox, rep*
$\mathcal{N}$		*ctx, dom(r), gap*
$\mathcal{O}$		*ctx, chunk(p), gap, rec*

(a) (continued)

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$
$\mathcal{A}$	$\mathcal{B}$	$\mathcal{B}$	$\mathcal{H}$	$\mathcal{B}$	$\mathcal{I}$		
$\mathcal{J}$	$\mathcal{K}$	$\mathcal{L}$					
$\mathcal{M}$	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{O}$				
$\mathcal{M}$	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{O}$				

(b)

A				B				B			
0	0	0	1	2	2	2	2	2	2	2	2
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Words	are	fly-	-ing	out	like	end-	-less	rain	in-	-to	a

C				D				E			
2	2	2	4	2	2	2	2	2	2	2	0
0.25	0.5	0.75	0.5	0.25	0.5	0.75	0.5	0.5	0.5	0.25	0.5
pa-	-per	cup	They	slith-	-er	while	they	pass	they	slip	a-

F		G					A			
2	2	2	2	2	2	$1^{\ddagger}$	0	0	0	1
0.75	0.5	0.5	0.5	0.25	0.5	$2.25^{\ddagger}$	0.5	0.5	0.5	0.5
-way	a-	-cross	the	u-	-ni-	verse	Pools	of	sor-	-row

B				B				H			
2	2	2	2	2	2	2	2	2	2	2	4
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.75	0.25
waves	of	joy	are	drift-	-ing	through	my	o-	-pen	mind	pos-

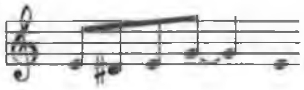
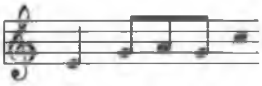
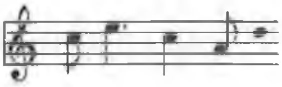
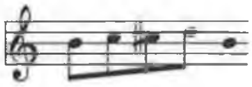
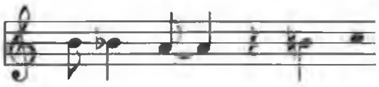
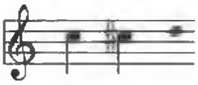
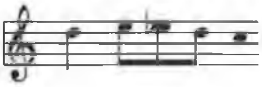
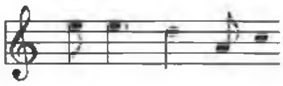
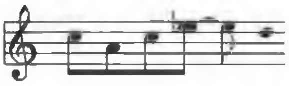
B				I				J			
2	2	2	2	1	2	2	7	3	3	5	1
0.5	0.5	0.5	0.5	0.5	0.5	0.5	2.5	0.5	0.5	0.5	1.0
-sess-	-ing	and	ca-	-ress-	-ing	me	—	Jai	—	Gu-	-ru

K		L			M				N		
3	2	0	5	7	0	0	0	2	2	9	$7^{\ddagger}$
1.0	0.5	1.5	2.5	4.0	0.5	0.5	0.5	0.5	1.0	0.5	$4.5^{\ddagger}$
—	De	—	va	Om	no-	thing's	gon-	-na	change	my	world

A				O			M			
0	0	0	1	1	8	$10^{\ddagger}$	0	0	0	2
0.5	0.5	0.5	0.5	1.0	0.5	$4.5^{\ddagger}$	0.5	0.5	0.5	0.5
No-	thing's	gon-	-na	change	my	world	No-	thing's	gon-	-na

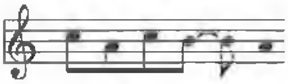
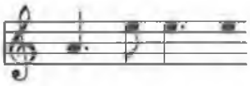
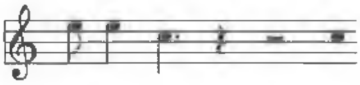
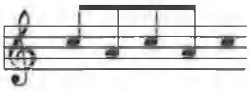
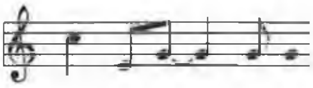
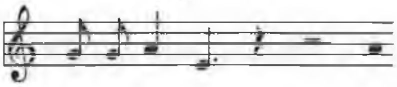
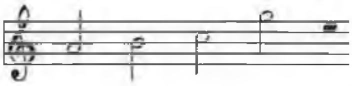
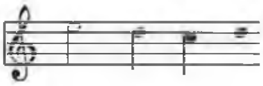
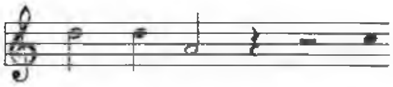
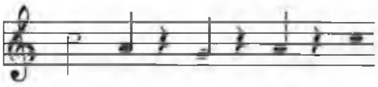
N			A				O		
2	9	$7^{\ddagger}$	0	0	0	1	1	8	‡
1.0	0.5	$4.5^{\ddagger}$	0.5	0.5	0.5	0.5	1.0	0.5	‡
change	my	world	No-	-thing's	gon-	-na	change	my	world

(c)

Label	Phrase	Factors
$\mathcal{A}$		*dom(r), gap, prim, prox, rep*
$\mathcal{B}$		*chunk, dom(r), gap, rep*
$\mathcal{C}$		*dom(r), gap, rep*
$\mathcal{D}$		*dom(r), prox, rep*
$\mathcal{E}$		*chunk, dom(r)*
$\mathcal{F}$		*alg*
$\mathcal{G}$		*chunk, dom(r), gap, rep*
$\mathcal{H}$		*dom(r), gap, rep*
$\mathcal{I}$		*chunk, dom(r), gap, prox, rep*

(a)

Figure 12.10: The boundary points discovered for the melody *When I'm Sixty Four* (see Appendix E.4.10). Where (a) illustrates the learned melodic phrases, (b) illustrates the segmentation formed and (c) illustrates the placement of each boundary point using the pitch-interval and IOI representations.

Label	Phrase	Factors
$\mathcal{J}$		*chunk, dom(r), gap, prox, rep*
$\mathcal{K}$		*alg*
$\mathcal{L}$		*alg*
$\mathcal{M}$		*dom(r), prox, rep*
$\mathcal{N}$		*chunk, dom(r), gap, rep*
$\mathcal{O}$		*alg*
$\mathcal{P}$		*dom(r), gap, rep*
$\mathcal{Q}$		*alg*
$\mathcal{R}$		*dom(r), gap*
$\mathcal{S}$		*dom(r), gap, rep*

(a) (continued)

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$		
$\mathcal{D}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$
$\mathcal{A}$	$\mathcal{B}$	$\mathcal{H}$		
$\mathcal{I}$	$\mathcal{J}$	$\mathcal{K}$	$\mathcal{L}$	
$\mathcal{M}$	$\mathcal{M}$	$\mathcal{M}$	$\mathcal{N}$	$\mathcal{O}$
$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{R}$	$\mathcal{S}$	

(b)

A				*B*					*C*				
1	1	3	3	3	2	2	5	0‡	4	4	3	5	3‡
0.333	0.167	0.333	0.667	0.5	0.333	0.167	0.333	1.17‡	0.25	0.75	0.5	0.333	2.17‡
When	I	get	old-	-er	los-	-ing	my	hair	ma-	-ny	years	from	now

D				*D*				*E*			
1	1	1	3	1	1	1	3	1	1	2	1
0.333	0.167	0.333	0.167	0.333	0.167	0.333	0.167	0.25	0.5	1.25	0.5
will	you	still	be	send-	-ing	me	a	va-	-len-	-tine	birth-

F		*G*					*A*			
1	1	2	1	1	2	8‡	1	1	3	3
0.5	0.5	0.5	0.333	0.167	0.333	1.17‡	0.333	0.167	0.333	0.667
-day	greet-	-ing	bot-	-tle	of	wine	If	I'd	been	out

B					*H*				
3	2	2	5	4‡	0	2	5	3	0‡
0.5	0.333	0.167	0.333	1.17‡	0.25	0.75	0.5	0.333	2.17‡
till	quar-	-ter	to	three	would	you	lock	the	door

I					*J*			
3	3	3	1	2‡	3	3	1	2
0.333	0.167	0.333	0.417	0.75‡	0.333	0.167	0.333	0.417
will	you	still	need	me	will	you	still	feed

K			*L*			*M*			
7	0	0	0	4	0	3	3	3	3
0.75	0.25	0.75	0.25	0.5	2.25	0.333	0.167	0.333	0.167
me	when	I'm	six-	-ty	four	ev-	-ry	sum-	-mer

M				*M*			
3	3	3	3	3	3	3	3
0.333	0.167	0.333	0.167	0.333	0.167	0.333	0.167
we	can	rent	a	cot-	-tage	in	the

N				*O*				*P*				
8	3	0	0	0	2	5	5	2	1	7	3	0‡
0.5	0.333	0.667	0.333	0.167	0.25	0.5	2.25	1.0	1.0	1.0	1.0	6.0‡
Isle	of	Wight	if	it's	not	too	dear	You'll	be	old-	-er	too

Q			*R*			*S*				
2	2	2	0	5	3	3	2	2	3	‡
1.0	0.5	0.5	1.0	0.5	2.5	1.0	1.0	1.0	1.0	‡
And	if	you	say	the	word	I	could	stay	with	you

(c)

12.5 Analysis

Proximity: Generally, the proximity of notes in terms of pitch and time contributed frequently to the formation of melodic phrases. Time in particular had a significant influence, not only in terms of closeness, but also in terms of distance. This is because relatively large gaps between notes almost always indicated a phrase boundary. Consequently, the rhythmic cues of temporal proximity and the gap principle were modelled well during the segmentation process. See phrases $\mathcal{D}$ and $\mathcal{F}$ from *Yesterday* for examples of temporal proximity and the gap principle. Pitch proximity also influenced the formation of phrases, but to a lesser extent (see phrase $\mathcal{B}$ from *Yesterday*).

Repetition: Repetition also played a vital role in the formation of melodic phrases. Rhythmic repetition was more prevalent in the melodies and frequently contributed to the formation of phrases, although pitch repetition did play a role (see phrase $\mathcal{B}$ from *Across the Universe*). Frequently, the repetition of phrases resulted in a strong indication of a melody's structure; such as a verse repetition or the beginning of a new section. See *Two of Us*, *Here Comes the Sun*, *Don't Let Me Down* and *Across the Universe* for good examples of this.

Context: The same pattern occurring in different contexts contributed the formation of melodic phrases. For rhythms, this often occurred (see phrases $\mathcal{E}$, $\mathcal{F}$ and $\mathcal{G}$ from *Yesterday*). Pitch sequences occurring in different contexts only sometimes influenced the segmentations formed (see phrase $\mathcal{H}$ from *Hey Jude*).

Chunking-out: Influences from previously learned patterns made a large contribution to the formation of phrases. When a pattern was chunked-out, subsequent patterns had an extended time period in STM before the $\mathcal{F}_1$ overload mechanism kicked in. This extended time period enabled these patterns to be learned with greater ease. See phrase $\mathcal{B}$ from *Here Comes the Sun*.

Primacy and recency effect: Primacy and recency effects were also contributing factors in the segmentation process. At the start of a melody, a period of seven attentional time-spans occurred at the pitch module and a period of six tactus-spans occurred at the rhythm module before their first input assembly was reset. This provided cell assemblies an extended time period to learn patterns at the beginning of a melody. Furthermore, due to the addition of seven extra beats at the end of each melody (see Section 12.2.5), an extended time period was also afforded to cell assemblies in order to learn phrases at the end of a melody.

Memory restrictions: The size of the phrases learned were restricted by the depth and capacity of STM (specified by K_{11}, K_{12} and K_{chunk}) and the limits imposed on the maximum sized chunk a cell assembly may learn (N_l). Consequently, the maximum sized pattern that could be formed consisted of five onsets, which is consistent with the transient memory span of human STM.

Dominance: During the segmentation process the dimension of rhythm was dominant. Although this condition was explicitly built into the segmentation process (see Section 12.2.6), preliminary SONNET-MAP simulations showed that, in general, this proved to be the case anyway. This is consistent with studies in melody perception which show that, in general, time dominates the perception of melodies.

Finally, although most melodic phrases were discovered using SONNET-MAP, the algorithm presented in Section 12.2.6, used to *fill in the gaps*, helped form some of the phrases. However, in most cases the sequences of notes that were not learned by SONNET-MAP consisted of five or less note onsets. As a consequence, a group could be directly formed from these onsets. In this sense, the major contributor to the formation of these patterns was in fact SONNET-MAP. This is because the boundaries of each phrase were specified by the preceding and subsequent phrases which brace the unlearned phrase.

12.6 Summary

In this chapter, the manner by which SONNET-MAP can be used to segment melodies was experimentally illustrated. The empirical setup and results were discussed in detail. In was found that, in general, the segmentations formed were consistent with many aspects of human melody perception. Using the boundary points discovered during these experiments, Chapter 13 will empirically show how *Re*TREE*ve* networks can be trained with the entire set of fifty *Beatles* melodies, and in turn utilized as a CBR system using the algorithm introduced in Chapter 11.

Chapter 13

Content-Based Retrieval of Melodies using *Re*TREE*ve*

13.1 Introduction

Chapter 11 introduced *Re*TREE*ve* and showed how it can be applied to melody retrieval. In order to illustrate that *Re*TREE*ve* functions as intended, this chapter presents empirical results which evaluate its retrieval performance on the *Beatles* test suite. In particular, this chapter demonstrates that:

- Queries that are aligned along phrase boundaries are retrieved more effectively than queries that are not aligned along phrase boundaries. In other words, fewer notes are required to achieve a correct retrieval. This indicates the benefits of melody segmentation and, in particular, pursuing human-based segmentation techniques. This is because phrases are more likely to coincide with the places in a melody that humans consider most memorable, and therefore the most likely places a query will begin. In this sense, the organization of melodies into human-inspired, multidimensional phrase hierarchies effectively reduces the retrieval search space. Consequently, not only does the accuracy and amount of information contained in a query affect retrieval performance, so too does the manner in which melody segmentation is performed.

- The combination of two dimensions (pitch and rhythm) significantly improves retrieval performance over the use of a single dimension (either pitch or rhythm). Using two dimensions yields significantly better retrieval accuracy and retrieval times for both perfect and imperfect queries. This is because two dimensions contain more information from which to facilitate the restriction of possible matches to a relatively small number of pitch-rhythm pairs. Consequently, the retrieval search space is markedly reduced.

- Retrieval performance degrades gracefully as the quality of the queries degrade. Hence, recognition performance is robust. This is achieved by varying the match, or tolerance parameter K_M to coincide with the level of query degradation permitted. Thus, phrases within imperfectly specified queries are more likely to pass the tolerance threshold and be considered during the template matching phase of the *Re*TREE*ve* algorithm.

- Particular melody representations are more suited to particular query types. For example, a pitch-interval representation is better suited to very noisy queries than an IPC representation. This is because, even if pitch information is badly specified in a query, recognition remains robust because the similarity between notes can be measured. Conversely, an IPC representation is more suited to perfect queries than pitch-intervals because more information is available from which to better discriminate between potential retrievals.

Before the empirical results are presented, the manner in which the *Re*TREE*ve* network was populated and its experimental setup will be discussed.

13.2 Populating *Re*TREE*ve* with *The Beatles* melodies

Chapter 10 introduced the SONNET-MAP network and specified how it can be utilized to automatically segment melodies. Subsequently, Chapter 12 described empirical results showing the segmentations SONNET-MAP formed using a selection of the Beatles melodies catalogued in Appendix E. These results illustrated the boundary points along which each melody is grouped into phrases. These boundary points are utilized to populate the *Re*TREE*ve* networks used in this chapter. Refer to Section 11.4 to see how this is achieved.

In this chapter, separate *Re*TREE*ve* networks are evaluated; one which processes IPC and IOI information, another which processes pitch-interval and IOI information and, finally, one which processes contour and IOI information. Although it is possible to combine each of the these features into a four-feature parallel network (see Section 14.3), here we evaluate each combination separately.

13.3 Experimental setup

13.3.1 Automatically generating the test queries

Once the *Re*TREE*ve* network is populated with melodies, it can be tested with melodic queries to evaluate its ability as a CBR system. *Re*TREE*ve* is tested using three different types of queries; *perfect queries*, *noisy queries* and *noisy queries with insertions and deletions.* Remember (see Chapter 3), these queries model the types of errors that humans could make when composing a query. This section describes how these queries are generated.

A set of queries is generated in the following way (see Table 13.1). Firstly, a set of perfect queries, of length Q_{length}, are generated from the entire set of *Beatles* melodies. This is achieved by selecting the first Q_{length} notes from each melody, then the second Q_{length} notes, and so forth until the final query of Q_{length} is selected. From this set, a number of queries are chosen that

Parameter	**Description**
Q_{bound}	The number of queries to be used which are aligned along boundary points.
Q_{nbound}	The number of queries to be used which are not aligned along boundary points.
Q_{length}	The length of each query.
ΔQ_{nmin}, ΔQ_{nmax}	The minimum and maximum number of notes to be transformed from the perfect query of length Q_{length}. The actual number is chosen randomly within this range.
ΔQ_{pmin}, ΔQ_{pmax}	If a note's pitch is to be altered, these parameters specify the minimum and maximum number of semitones by which the pitch should be changed. The actual number of semitones to be changed is chosen randomly within this range. Furthermore, there is a 50% chance that the pitch is either increased or decreased.
ΔQ_{rmin}, ΔQ_{rmax}	If a note's IOI is to be altered, this parameter specifies the minimum and maximum percentage (in terms of the IOI) by which the IOI should be changed. The actual percentage is chosen randomly within this range. Furthermore, there is a 50% chance that the change either increases or decreases the length of the IOI.
ΔQ_{imin}, ΔQ_{imax}	Specifies the minimum and maximum number of insertions to be made. The actual number of insertions is chosen randomly within this range.
ΔQ_{dmin}, ΔQ_{dmax}	Specifies the minimum and maximum number of deletions to be made. The actual number of deletions is is chosen randomly within this range.

Table 13.1: Parameters used to generate test queries.

are aligned along boundary points (Q_{bound}) and a number are chosen that are not aligned along boundary points (Q_{nbound}). These are the queries to be transformed and used to test retrieval performance. Next, a set of note indices, ranging from zero to $Q_{length} - 1$ is generated, which is randomly shuffled. Then, the first ΔQ_{nmax} indices from this set are used to identify the set of notes to be altered according to the parameters specified in Table 13.1. For example, Figure 13.1 shows two queries generated as described above. Both queries are generated from the first twenty notes of the melody *Here Comes The Sun.* Figure 13.1(a) illustrates an example of a perfect query, whereas Figure 13.1(b) illustrates an example of a noisy query. The settings used to generate all of the queries used in this chapter are summarized in Table 13.2.

IPC	4	0	2	4	4	2	0	7	9	0	9	4	2	0	9	7	4	0	2	4
IOI	0.5	0.5	0.5	2.0	1.0	1.0	0.5	0.5	1.0	0.5	0.5	0.5	0.5	0.5	1.5	4.5	0.5	0.5	0.5	2.0

(a) Perfect query.

IPC	4	0	6	4	7	2	0	9	9	0	9	4	2	10	9	7	0	1	4	4
IOI	0.5	0.5	0.5	2.0	1.0	1.0	0.4	0.58	1.0	0.5	0.5	0.5	0.5	0.6	1.17	4.5	0.5	0.5	0.58	2.0

(b) Noisy query.

Figure 13.1: This is an example of a perfect query and a noisy query derived from the melody *Here Comes The Sun.* In these examples $Q_{length} = 20$ and the notes originate from the beginning of the melody. (a) This query yielded a correct retrieval after the presentation of just four notes. (b) The parameters that generated this query are as follows: $\Delta Q_{nmin} = 5$, $\Delta Q_{nmax} = 10$, $\Delta Q_{pmin} = 1$, $\Delta Q_{pmax} = 4$, $\Delta Q_{rmin} = 10\%$, $\Delta Q_{rmax} = 25\%$. No insertions nor deletions are present. This query yielded a correct retrieval after the presentation of thirteen notes. Furthermore, any note can have both its pitch and corresponding IOI changed.

	Perfect	**Noisy**			**Insert/Delete**
Tables	**13.5 – 13.11**	**13.12 – 13.14**	**13.15, 13.17, 13.19**	**13.16, 13.18, 13.20**	**13.21 – 13.23**
Q_{bound}	100 – 180	180	180	180	180
Q_{nbound}	20 – 100	20	20	20	20
Q_{length}	20	20	20	20	20
ΔQ_{nmin}	0	2 – 8	2	5	2
ΔQ_{nmax}	0	2 – 8	5	10	5
ΔQ_{pmin}	0	0	1	1	1
ΔQ_{pmax}	0	0	4	4	4
ΔQ_{rmin}	0	10% – 40%	10%	10%	10%
ΔQ_{rmax}	0	10% – 40%	25%	25%	25%
ΔQ_{imin}	0	0	0	0	1
ΔQ_{imax}	0	0	0	0	2
ΔQ_{dmin}	0	0	0	0	1
ΔQ_{dmax}	0	0	0	0	2

Table 13.2: Parameter values used to generate the test queries.

13.3.2 Architectural configuration

As described in Chapter 11, the architecture of *Re*TREE*ve* mirrors that of the SONNET-MAP network. In this chapter the architectural configuration of *Re*TREE*ve* is described in Table 13.3.

	IPC	PI	Contour	IOI
N_g	12	1	1	1
N_s	20	20	20	20
N_2	1000			
N_3	50			

Table 13.3: The architectural configuration of the *Re*TREE*ve* network utilized in this chapter. Note that the number of siblings (N_s) is always at least as large as Q_{length}, which is equal to twenty. This ensures that there is always adequate working memory to represent each melodic feature.

13.3.3 Parameter settings

This sections outlines the small number of parameters required for the operation of *Re*TREE*ve.* The parameter of primary importance is that of K_M. The lower K_M the greater the number of phrases that will match. Conversely, the higher K_M the smaller the number of phrase that will match. Consequently, to enable noisy queries to match, K_M should generally be set relatively low. If queries are fairly well preserved versions of the target melody, K_M should be set relatively high.

	IPC	PI	Contour	IOI
K_{encode}	MSG	CAR	CAR	CAR
ω	2	3	2	3
$K_{intervals}$	N/A	13	3	N/A
$K_{overlap}$	1			
K_M	0.8 – 0.99			

Table 13.4: The parameter values utilized by the *Re*TREE*ve* network.

13.3.4 Interpreting the results

Each query is presented to *Re*TREE*ve* once. After the presentation of each note, the *Re*TREE*ve* algorithm will return a ranked list containing the top five best matches. We have chosen to consider the top five ranks rather than just the top ranked match for a number of reasons. Firstly, it sometimes transpires that some of the rank 1 results have identical match values to the rank 2, 3, 4 or 5 results. Consequently, although there may be a tie, the desired retrieval may in fact have occurred. Secondly, it is general practice to issue a list of possible matches (for example, internet search engines), as in many cases (particularly with badly degraded imperfect queries) it is unlikely that perfect retrievals will occur at rank 1.

Each set of query results are presented in a table which specifies the percentage of correct retrievals at a particular rank at a particular note. For example, in Table 13.5, 99% of the queries, which are aligned on boundary points, are correctly identified as one of the top 2 ranked melodies by the 5^{th} note. Remember, each test utilized 200 queries ($Q_{bound} + Q_{nbound}$). Furthermore, unless explicitly noted, when a particular pitch representation is mentioned it should be assumed that it is coupled with the use of the IOI representation. This convention is employed for the sake of convenience.

13.4 Retrieval results

13.4.1 Perfect queries

	Boundary (%)					Non-boundary (%)				
Rank	1	2	3	4	5	1	2	3	4	5
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	36.00	38.00	38.00	38.00	38.00	1.00	1.00	1.00	1.00	1.00
4	59.00	61.00	61.00	61.00	61.00	21.00	21.00	23.00	23.00	23.00
5	96.00	99.00	**100.00**	100.00	100.00	35.00	35.00	**37.00**	37.00	37.00
6	96.00	99.00	100.00	100.00	100.00	62.00	64.00	65.00	65.00	65.00
7	99.00	99.00	100.00	100.00	100.00	83.00	85.00	85.00	85.00	85.00
8	100.00	100.00	100.00	100.00	100.00	94.00	97.00	97.00	97.00	97.00
9	100.00	100.00	100.00	100.00	100.00	98.00	100.00	100.00	100.00	100.00
10	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
11	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
12	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
13	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
15	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 13.5: Aggregated retrieval results for *IPC's and IOI's* using perfect queries. In this table a distinction is made between queries that are aligned along boundary points and queries that are not aligned along boundary points. In this experiment, 100 boundary queries and 100 non-boundary queries were tested. It is clear from this table that queries which are aligned along boundary points are retrieved significantly quicker. For example, by the 5^{th} note, 100% of queries are correctly identified as being taken from one of the top 3 ranked melodies if they are aligned along boundary points. On the other hand, queries that are not aligned along boundary points are only retrieved correctly 37% of the time by the 5^{th} note.

	Boundary (%)					**Non-boundary (%)**				
Rank	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
3	32.00	36.00	37.00	40.00	40.00	1.00	1.00	2.00	2.00	2.00
4	54.00	57.00	60.00	64.00	64.00	16.00	20.00	21.00	23.00	24.00
5	88.00	94.00	**97.00**	99.00	100.00	26.00	31.00	**35.00**	38.00	39.00
6	90.00	96.00	98.00	99.00	99.00	51.00	55.00	61.00	63.00	65.00
7	95.00	98.00	98.00	99.00	99.00	75.00	77.00	82.00	83.00	84.00
8	96.00	100.00	100.00	100.00	100.00	89.00	91.00	94.00	95.00	95.00
9	100.00	100.00	100.00	100.00	100.00	94.00	96.00	97.00	98.00	98.00
10	100.00	100.00	100.00	100.00	100.00	98.00	100.00	100.00	100.00	100.00
11	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
12	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
13	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
15	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 13.6: Aggregated retrieval results for *pitch-intervals and IOI's* using perfect queries.

	Boundary (%)					Non-boundary (%)				
Rank	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	28.00	36.00	41.00	41.00	41.00	2.00	2.00	3.00	3.00	3.00
4	43.00	53.00	58.00	63.00	63.00	15.00	18.00	21.00	23.00	26.00
5	78.00	88.00	**95.00**	98.00	100.00	18.00	28.00	**33.00**	37.00	38.00
6	84.00	92.00	95.00	98.00	99.00	43.00	49.00	56.00	60.00	64.00
7	92.00	96.00	96.00	98.00	98.00	65.00	71.00	75.00	77.00	81.00
8	94.00	99.00	99.00	100.00	100.00	80.00	87.00	89.00	90.00	94.00
9	100.00	100.00	100.00	100.00	100.00	84.00	90.00	91.00	95.00	96.00
10	100.00	100.00	100.00	100.00	100.00	93.00	96.00	96.00	99.00	99.00
11	100.00	100.00	100.00	100.00	100.00	96.00	98.00	98.00	99.00	100.00
12	100.00	100.00	100.00	100.00	100.00	99.00	100.00	100.00	100.00	100.00
13	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
15	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 13.7: Aggregated retrieval results for *contours and IOI's* using perfect queries.

	Rank (%)				
Note	1	2	3	4	5
1	0.00	0.00	0.00	0.00	0.00
2	1.00	1.50	1.50	1.50	1.50
3	25.50	31.50	36.00	36.50	36.50
4	39.00	44.50	48.50	53.00	53.50
5	69.50	82.00	87.50	91.00	91.00
6	73.00	87.50	91.00	93.50	93.50
7	81.50	91.50	94.00	97.00	97.50
8	89.50	97.00	98.00	100.00	100.00
9	96.50	98.50	98.50	100.00	100.00
10	97.50	99.50	99.50	100.00	100.00
11	98.50	99.00	99.50	100.00	100.00
12	99.50	100.00	100.00	100.00	100.00
13	100.00	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00
15	100.00	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00

Table 13.8: Aggregated retrieval results for *IPC's* using perfect queries. In this experiment and the remaining experiments described in this chapter, 90% of test queries will be aligned along phrase boundaries and 10% will not be aligned along phrase boundaries. This is an attempt to model more accurately the types of queries a human might produce. Note also, that the remaining experiments in this section only use a single melodic feature. As a consequence, retrieval performance is poorer than when two features are utilized.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	2.00	2.50	2.50	2.50	3.00
3	10.50	18.50	27.50	34.50	39.50
4	13.50	21.50	25.50	32.50	39.50
5	40.00	59.00	65.50	73.00	77.50
6	41.50	55.50	63.00	71.50	77.50
7	50.00	66.00	75.00	79.00	84.00
8	66.50	78.00	84.00	87.50	93.00
9	80.00	89.50	94.00	95.50	96.50
10	88.50	94.50	96.50	98.00	98.50
11	92.00	97.00	98.00	98.00	99.00
12	93.50	97.50	98.50	99.50	100.00
13	99.00	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00
15	99.50	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00

Table 13.9: Aggregated retrieval results for *pitch-intervals* using perfect queries.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	2.50	4.50	7.00	7.00	7.00
3	2.50	5.00	11.00	18.00	21.00
4	4.50	10.50	15.50	18.00	22.00
5	15.00	27.50	35.00	44.00	53.50
6	13.50	25.00	32.50	38.00	46.50
7	24.00	36.50	46.00	53.50	57.00
8	31.00	46.00	53.50	61.00	64.50
9	48.00	60.50	72.50	80.00	83.50
10	57.50	66.50	81.00	87.50	90.50
11	63.00	78.50	84.00	89.00	92.00
12	65.00	81.50	90.50	93.00	94.50
13	75.50	89.50	94.00	96.50	97.50
14	83.50	95.00	96.50	99.00	99.00
15	91.50	96.50	97.50	99.00	99.50
16	90.00	95.50	98.50	99.00	100.00
17	91.00	97.50	100.00	100.00	100.00
18	95.00	99.00	99.50	100.00	100.00
19	97.00	99.00	99.50	100.00	100.00
20	98.00	99.50	100.00	100.00	100.00

Table 13.10: Aggregated retrieval results for *contours* using perfect queries.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	3.50	4.50	4.50	5.50	5.50
3	14.50	22.50	25.50	28.00	31.00
4	18.50	28.00	34.50	39.00	39.50
5	44.00	56.00	68.00	70.50	71.50
6	48.00	58.50	64.00	71.00	74.50
7	58.00	73.50	78.50	83.50	85.00
8	68.50	81.00	83.00	88.00	92.00
9	76.50	87.50	91.50	93.00	94.00
10	81.50	89.50	91.50	94.00	94.50
11	87.00	93.00	96.00	97.00	97.50
12	89.50	95.50	96.50	98.00	98.50
13	93.50	98.00	99.00	99.00	99.00
14	93.00	98.00	100.00	100.00	100.00
15	96.00	99.50	100.00	100.00	100.00
16	96.50	99.00	100.00	100.00	100.00
17	99.50	99.50	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00
19	99.50	100.00	100.00	100.00	100.00
20	99.50	100.00	100.00	100.00	100.00

Table 13.11: Aggregated retrieval results for *IOI's* using perfect queries.

13.4.2 Noisy IOI's — graceful degradation and associative priming

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00
3	30.50	31.00	31.50	31.50	31.50
4	52.50	53.00	53.50	53.50	53.50
5	85.50	91.00	91.50	92.00	92.00
6	90.00	95.00	95.00	95.00	96.00
7	95.50	98.00	98.00	98.00	99.00
8	99.00	99.50	99.50	99.50	100.00
9	99.00	99.50	99.50	99.50	100.00
10	100.00	100.00	100.00	100.00	100.00
11	100.00	100.00	100.00	100.00	100.00
12	100.00	100.00	100.00	100.00	100.00
13	100.00	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00
15	100.00	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00

Table 13.12: Aggregated retrieval results for *perfect IPC's and noisy IOI's.* In this experiment, the IOI of two notes are degraded by 10% and $K_M = 0.9$. Despite this, by the 10^{th} note at rank 1, 100% accuracy is achieved.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00
3	31.00	31.00	31.50	31.50	31.50
4	53.00	54.00	54.50	54.50	54.50
5	84.50	89.00	90.00	90.50	90.50
6	88.50	94.00	95.00	95.00	95.50
7	92.00	95.50	96.50	97.00	97.50
8	96.50	99.00	99.50	99.50	100.00
9	97.50	99.00	99.50	99.50	100.00
10	99.00	100.00	100.00	100.00	100.00
11	99.50	100.00	100.00	100.00	100.00
12	99.50	100.00	100.00	100.00	100.00
13	99.50	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00
15	100.00	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00

Table 13.13: Aggregated retrieval results for *perfect IPC's and noisy IOI's.* In this experiment, the IOI of four notes are degraded by 20% and $K_M = 0.85$. Despite this, by the 10^{th} note at rank 1, 99% accuracy is achieved.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00
3	27.00	29.00	30.00	30.00	30.00
4	43.50	46.00	48.00	48.50	48.50
5	69.00	75.00	76.00	77.50	77.50
6	73.50	80.50	82.50	83.00	83.00
7	80.50	87.00	90.00	90.50	90.50
8	85.50	90.00	92.50	93.50	93.50
9	90.00	94.00	97.00	97.50	97.50
10	90.00	94.00	98.00	98.50	98.50
11	92.50	95.50	98.50	98.50	98.50
12	93.50	96.00	98.50	98.50	98.50
13	96.50	98.00	99.00	99.00	99.00
14	97.00	98.00	99.00	99.00	99.00
15	98.00	99.00	100.00	100.00	100.00
16	97.50	99.00	100.00	100.00	100.00
17	97.00	99.00	100.00	100.00	100.00
18	97.50	99.00	100.00	100.00	100.00
19	97.50	99.00	99.50	99.50	100.00
20	97.50	99.00	99.50	99.50	100.00

Table 13.14: Aggregated retrieval results for *perfect IPC's and noisy IOI's.* In this experiment, the IOI of eight notes are degraded by 40% and $K_M = 0.8$. Despite this, by the 10^{th} note at rank 1, 90% accuracy is achieved.

13.4.3 Noisy queries

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	0.50	0.50	0.50	0.50	0.50
3	16.50	16.50	16.50	16.50	16.50
4	27.00	27.00	27.00	27.00	27.00
5	43.50	44.50	44.50	44.50	44.50
6	50.00	51.50	51.50	51.50	51.50
7	55.50	56.50	57.50	57.50	57.50
8	61.00	62.50	63.50	63.50	63.50
9	69.50	71.50	72.00	72.00	72.50
10	74.50	77.00	78.00	78.00	78.50
11	80.00	82.50	83.50	83.50	84.00
12	81.00	84.00	85.00	85.00	85.50
13	82.50	86.00	87.00	87.00	87.00
14	85.00	88.50	89.50	89.50	89.50
15	87.00	90.00	91.00	91.00	91.00
16	88.50	92.00	93.50	93.50	93.50
17	89.00	93.00	94.50	95.00	95.00
18	89.50	94.00	95.50	96.00	96.00
19	90.50	95.50	96.50	96.50	96.50
20	91.50	96.50	97.50	97.50	97.50

Table 13.15: Aggregated retrieval results for *noisy IPC's and noisy IOI's.* In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.99$ and $K_M^b = 0.9$. The queries in this experiment are significantly degraded; nevertheless, 97.5% of queries were correctly retrieved by the 20th note at rank 5.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00
3	9.00	9.00	9.00	9.00	9.00
4	12.00	12.50	12.50	12.50	12.50
5	14.50	16.00	16.00	16.00	16.00
6	16.50	18.50	18.50	18.50	18.50
7	20.50	22.00	22.50	22.50	22.50
8	24.50	25.50	26.00	26.00	26.00
9	27.50	29.50	30.00	30.00	30.00
10	31.50	33.00	33.50	33.50	33.50
11	34.00	36.00	36.50	36.50	36.50
12	35.50	38.00	38.50	38.50	38.50
13	38.50	42.00	43.00	43.00	43.00
14	41.00	44.50	46.00	46.00	46.00
15	44.00	47.50	49.00	49.00	49.00
16	44.50	49.00	50.50	50.50	50.50
17	45.00	50.50	52.00	52.50	52.50
18	46.50	52.00	53.50	54.00	54.00
19	48.50	54.00	56.00	56.50	56.50
20	50.50	56.50	58.50	59.00	59.00

Table 13.16: Aggregated retrieval results for *noisy IPC's and noisy IOI's*. In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.99$ and $K_M^b = 0.9$. The queries in this experiment are highly degraded, with 59% of queries being correctly retrieved by the 20^{th} note at rank 5.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	0.50	0.50	0.50	0.50	0.50
3	16.50	17.00	18.00	18.00	18.00
4	26.00	27.50	29.00	30.00	30.00
5	38.00	44.00	46.50	47.50	47.50
6	41.00	49.00	52.50	54.00	55.00
7	44.00	53.50	57.00	59.00	61.00
8	48.50	59.00	62.00	64.50	67.50
9	56.50	65.50	69.00	70.00	73.00
10	61.00	68.50	72.50	75.00	77.50
11	65.00	74.00	78.00	79.00	81.50
12	68.00	76.00	80.00	81.00	83.50
13	71.00	79.00	81.50	83.00	85.50
14	74.50	82.00	84.00	86.50	88.00
15	78.00	84.50	87.50	89.00	90.50
16	79.00	85.50	88.50	90.50	92.00
17	79.00	86.00	89.00	91.00	92.00
18	78.00	87.00	89.00	91.50	92.50
19	78.50	88.50	90.50	91.50	92.50
20	79.00	89.50	91.50	93.00	94.50

Table 13.17: Aggregated retrieval results for *noisy pitch intervals and noisy IOI's.* In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.99$ and $K_M^b = 0.9$. The queries in this experiment are significantly degraded; nevertheless, 94.5% of queries were correctly retrieved by the 20^{th} note at rank 5.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	1.00	5.00	5.50	6.00	6.00
3	7.00	15.00	21.50	27.00	28.00
4	7.00	14.50	20.50	26.00	30.50
5	15.50	21.50	30.00	33.50	38.50
6	14.00	24.50	29.00	31.00	37.00
7	24.00	35.50	39.50	45.00	49.50
8	25.50	38.00	41.00	45.00	50.50
9	28.50	40.50	47.00	52.00	57.50
10	36.00	45.00	51.00	57.00	63.00
11	36.50	49.50	54.50	60.50	64.00
12	35.50	46.00	55.00	60.50	66.00
13	36.50	48.00	57.50	64.00	68.00
14	36.50	48.50	57.00	62.00	67.00
15	39.50	50.50	60.50	65.50	70.00
16	42.50	55.50	63.00	67.50	72.00
17	44.00	56.50	64.50	68.00	72.50
18	46.00	58.50	66.00	70.00	74.50
19	47.00	60.00	67.00	73.00	77.00
20	50.50	59.50	69.00	76.50	80.50

Table 13.18: Aggregated retrieval results for *noisy pitch intervals and noisy IOI's*. In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.85$ and $K_M^b = 0.85$. The queries in this experiment are highly degraded, with 80.5% of queries being correctly retrieved by the 20^{th} note at rank 5. This represents a significant improvement in performance compared with the use of IPC's for the same degraded query set.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	2.00	2.50	2.50	2.50	2.50
3	18.50	24.50	28.50	29.00	29.50
4	23.50	32.00	37.00	41.00	43.50
5	38.00	52.00	59.00	65.00	65.50
6	38.00	49.50	60.50	65.00	70.00
7	43.00	53.50	62.50	66.50	70.00
8	47.00	58.50	67.00	70.50	73.50
9	58.00	66.00	72.50	74.00	76.50
10	61.50	67.00	75.50	78.00	81.00
11	62.00	69.50	79.50	82.50	85.50
12	65.00	71.50	80.00	84.50	86.00
13	67.00	75.50	80.00	86.50	87.50
14	67.00	76.00	82.50	87.00	88.00
15	71.00	77.00	85.50	89.00	89.50
16	73.00	80.50	85.00	88.50	91.50
17	75.00	83.50	87.00	88.50	91.50
18	75.00	84.50	87.50	89.00	91.00
19	78.00	85.00	87.00	89.00	91.00
20	80.50	86.00	89.00	90.50	93.00

Table 13.19: Aggregated retrieval results for *noisy contours and noisy IOI's.* In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.99$ and $K_M^b = 0.9$. The queries in this experiment are significantly degraded; nevertheless, 93% of queries were correctly retrieved by the 20^{th} note at rank 5.

	Rank (%)				
Note	1	2	3	4	5
1	0.00	0.00	0.00	0.00	0.00
2	1.00	1.50	1.50	2.00	2.00
3	11.00	18.00	21.50	23.00	23.50
4	14.00	20.50	25.00	29.00	31.50
5	21.50	31.00	37.00	42.00	43.50
6	20.50	30.00	35.50	42.00	44.00
7	24.50	34.00	40.50	45.00	47.50
8	24.00	38.00	45.00	49.00	52.50
9	30.00	40.00	48.50	53.00	56.50
10	33.50	43.00	52.50	58.00	62.00
11	35.50	44.00	53.50	60.00	64.50
12	38.50	47.00	55.00	61.00	65.50
13	41.00	48.00	56.50	62.00	65.50
14	42.00	49.00	56.00	63.50	68.50
15	43.50	51.50	56.00	66.00	68.50
16	43.50	51.50	57.50	63.00	68.50
17	43.00	51.50	59.00	64.00	71.00
18	43.50	52.50	60.50	66.00	70.00
19	45.00	55.50	61.00	68.00	72.00
20	45.50	56.00	63.50	70.50	73.50

Table 13.20: Aggregated retrieval results for *noisy contours and noisy IOI's*. In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.99$ and $K_M^b = 0.9$. The queries in this experiment are highly degraded, with 73.5% of queries being correctly retrieved by the 20^{th} note at rank 5.

13.4.4 Queries with insertions and deletions

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	0.50	0.50	0.50	0.50	0.50
3	17.50	18.00	18.00	18.00	18.00
4	30.00	30.50	30.50	30.50	30.50
5	42.50	44.00	44.00	44.00	44.00
6	46.50	49.00	49.00	49.00	49.00
7	51.00	53.50	53.50	53.50	53.50
8	55.00	58.00	58.50	58.50	58.50
9	60.00	63.00	63.50	63.50	63.50
10	65.00	67.50	68.00	68.00	68.00
11	68.50	72.00	72.00	72.00	72.00
12	71.00	75.00	75.00	75.00	75.00
13	73.50	77.50	78.50	78.50	78.50
14	76.50	81.00	81.00	82.00	82.00
15	79.50	85.50	86.00	86.50	86.50
16	80.50	87.50	88.00	89.00	89.00
17	84.00	89.50	90.00	91.00	91.00
18	87.50	92.50	93.00	94.00	94.00
19	88.50	93.00	93.50	94.50	94.50
20	75.50	78.50	79.00	79.50	79.50

Table 13.21: Aggregated retrieval results for *noisy IPC's and noisy IOI's with insertions and deletions*. In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.99$ and $K_M^b = 0.9$. The queries in this experiment are highly degraded, nevertheless 88.5% of queries were correctly retrieved by the 19^{th} note at rank 1. Note that there is a discrepancy between the performance at rank 19 and rank 20. This discrepancy is due to the fact that some queries may have less than twenty notes due to deletions.

	Rank (%)				
Note	**1**	**2**	**3**	**4**	**5**
1	0.00	0.00	0.00	0.00	0.00
2	0.50	0.50	0.50	0.50	0.50
3	16.50	18.50	20.00	20.50	20.50
4	27.00	29.50	32.00	33.00	33.50
5	39.50	43.00	45.50	46.50	47.00
6	41.50	46.50	49.50	51.00	52.00
7	46.00	50.50	53.00	55.00	56.00
8	50.00	54.00	58.00	60.00	61.00
9	55.00	59.50	61.50	64.50	66.50
10	60.00	64.50	66.50	69.50	71.00
11	65.50	69.50	71.00	74.00	74.50
12	62.00	69.50	73.50	76.50	77.50
13	64.50	72.50	76.50	79.00	80.00
14	67.50	75.50	78.50	80.50	82.00
15	70.00	78.50	83.00	85.00	86.50
16	71.50	80.50	84.50	87.50	89.00
17	73.00	84.00	86.50	90.00	91.50
18	75.00	84.00	89.00	90.50	93.00
19	74.50	84.00	90.00	91.00	93.00
20	64.50	72.00	76.50	77.00	78.50

Table 13.22: Aggregated retrieval results for *noisy pitch intervals and noisy IOI's with insertions and deletions.* In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.99$ and $K_M^b = 0.9$. The queries in this experiment are highly degraded, nevertheless 74.5% of queries were correctly retrieved by the 19^{th} note at rank 1.

	Rank (%)				
Note	1	2	3	4	5
1	0.00	0.00	0.00	0.00	0.00
2	2.50	3.00	3.00	3.00	3.00
3	13.50	20.00	24.50	26.50	27.00
4	24.00	32.50	39.00	41.50	42.00
5	36.50	47.00	53.00	57.00	58.00
6	36.50	46.50	52.50	59.50	61.50
7	43.00	52.50	55.00	60.50	64.50
8	46.50	54.00	61.00	66.50	70.50
9	49.00	57.00	62.00	68.50	72.50
10	54.00	62.00	67.00	70.50	73.00
11	57.00	65.00	70.50	74.50	77.00
12	58.50	66.50	71.50	75.50	79.00
13	59.00	68.50	71.50	75.00	80.50
14	61.00	69.00	74.00	78.50	82.50
15	65.50	72.50	79.50	81.50	85.00
16	65.00	74.50	80.50	83.00	87.50
17	67.50	75.00	82.50	84.50	88.50
18	70.50	77.00	82.00	85.00	90.00
19	70.50	78.50	83.00	86.50	91.50
20	57.50	65.50	70.00	72.00	75.50

Table 13.23: Aggregated retrieval results for *noisy contours and noisy IOI's with insertions and deletions.* In addition to the parameter values already specified in Table 13.2, $K_M^a = 0.99$ and $K_M^b = 0.9$. The queries in this experiment are highly degraded, nevertheless 70.5% of queries were correctly retrieved by the 19^{th} note at rank 1.

13.5 Analysis

Perfect queries

The experimental results presented in Tables 13.5 to 13.11 illustrate *Re*TREE*ve*'s performance at retrieving perfect queries. Regarding these experiments there are a number of points to make:

- Tables 13.5, 13.6 and 13.7 illustrate that phrases which are aligned along phrase boundaries are retrieved significantly quicker than phrases which are not aligned along phrase boundaries. Overall retrieval performance for perfect queries is excellent.

- By comparing the results shown in Tables 13.5, 13.6 and 13.7 (two melodic features) against the results shown in Tables 13.8, 13.9, 13.10 and 13.11 (one melodic feature), it can clearly be seen that the use of both pitch and rhythm information during retrieval improves performance over the use of either pitch or rhythm information alone.

- Results show that, for perfect queries, the use of the IPC representation achieves the most accurate results, compared with the pitch-interval and melodic contour representations. This is because IPCs specify more information and thus provide more discriminatory power from which to yield more accurate results.

Noisy queries — graceful degradation and associative priming

The experimental results presented in Tables 13.12, 13.13 and 13.14 refer to queries whose rhythmic content was gradually degraded. These experiments illustrate that *Re*TREE*ve* 's performance degrades gracefully as queries become more degraded. This is due to the use of the match parameter K_M and the manner by which the IPC module can influence the IOI module. Additionally, although the IOI content of each query is degraded in these experiments, retrieval performance is still better than in the case where the IOI representation was solely used (see Table 13.11).

Noisy queries

Tables 13.15, 13.17, 13.19 presented the experimental results for queries which were significantly degraded. Nevertheless, performance remained robust. In particular, the IPC representation outperformed the pitch-interval and contour representations. This was due to the fact that enough pitch information still remained to allow the IPC representation to significantly influence recognition and yield good results.

Tables 13.16, 13.18, 13.20 presented the experimental results for queries which were highly degraded. Overall, retrieval performance for the pitch-interval representation proved most accurate.

Noisy queries with insertions and deletions

Finally, Tables 13.21, 13.22 and 13.23 illustrated retrieval performance using noisy queries with insertions and deletions. These types of queries are particularly difficult to process. However, once again, retrieval performance remained robust. In particular, Table 13.21 shows that enough pitch information still remained to allow the IPC representation to significantly influence recognition and yield good results.

13.6 Summary

This chapter presented the experimental results obtained using *Re*TREE*ve* on a wide range of query types: *perfect queries*, *noisy queries* and *noisy queries with insertions and deletions*. It was shown that queries which are aligned along boundary points are retrieved faster than queries which are not aligned along boundary points. Moreover, using multidimensional configurations of *Re*TREE*ve* outperformed single dimensional configurations. Also, retrieval performance degraded gracefully as noise levels increased. Finally, different representations were more suited to different types of queries. Generally, the IPC representation coupled with the IOI representation outperformed all other representations, except in the case when very noisy queries were presented. In these circumstances, the pitch-interval representation coupled with the IOI representation performed best.

Chapter 14

Conclusions

14.1 Introduction

This thesis introduced SONNET-MAP, which is a self-organizing neural network that can process multidimensional input sequences autonomously, in real-time and in real-world environments. In this thesis, SONNET-MAP has been applied to the problem of how melodies can be segmented using human-based perceptual cues. The boundary points discovered using SONNET-MAP were utilized to populate the *Re*TREE*ve* network introduced in Chapter 11. Using *Re*TREE*ve*, melodies can be retrieved in an efficient, robust and accurate manner using melody fragments taken from the original melody, which may even be inaccurate reproductions. As far as we are aware, SONNET-MAP and *Re*TREE*ve* constitute the first self-organizing neural network system to be applied to the problem of retrieving melodies by content.

This chapter synopsizes the major contributions of this thesis and suggests avenues of further investigation for SONNET-based research, melody segmentation and retrieval, applications in music processing and to other disciplines.

14.2 Contributions of this thesis

14.2.1 Innovations in representing melodies

Generally, in terms of short-term and long-term storage, this thesis challenges many common assumptions regarding the representation of melodies in CBR systems. Firstly, Chapter 9 identified different methods of representing the pitch and rhythmic dimensions of a melody in STM using cell activations. Furthermore, we discussed how the organization of parallel input fields can facilitate the parallel nature of LTM desired by SONNET-MAP and *Re*TREE*ve*. To this end, the following introductions were made in this thesis:

- A shared event field at $\mathcal{F}_0$ was introduced, which temporarily records the notes of a melody as they are presented. Each $\mathcal{F}_0$ cell represents a CPH, which remains active for the entire duration of each IOI.

- Methods were presented that show how certain characteristics of rhythm (IOI) and pitch (IPC, pitch-interval and melodic contour) can be derived from $\mathcal{F}_0$ and represented at $\mathcal{F}_1$ (using a variety of gating mechanisms). In particular, a new method of representing pitch-intervals and melodic contours was introduced. This method, which is similar to how Roberts represented IOIs using CARs, enables a direct comparison of pitches in STM with pitches stored in a cell assembly's LTM template. Currently, this is not possible using a GPC representational scheme, which most previous SONNET research focussed on. This new ability to measure pitch similarity proved advantageous for segmenting melodies because grouping by pitch proximity could be used as a grouping cue. Furthermore, the imperfect query tests described in Chapter 13 showed that this representation enabled retrieval results to be improved when very degraded queries were presented to *Re*TREE*ve*.
- A generalized and simplified $\mathcal{F}_1$ equation was introduced in Section 9.4.1. This equation worked extremely efficiently and its generality enables any melodic feature to be represented at $\mathcal{F}_1$. The only difference between each representation was the calculation of the term λ. An input assembly overload scheme based on activation time rather than activation levels was also introduced, which enabled the new input cell dynamics to operate correctly.

Secondly, in addition to the short-term storage of melodies in working memory, new work was also introduced which organizes a melody as a learned hierarchy of automatically segmented phrases, with two parallel SONNET modules representing the pitch and rhythm sequences independently. This representation mirrors the concept that many researchers now believe about the organization of melodies in the human brain (for example, the dissociation between different musical functions such as pitch and rhythm). This structural organization of melodies, utilized by both SONNET-MAP and *Re*TREE*ve*, is the first time such a structure has been used in either SONNET-based research or in CBR-based research. Indeed, such an organization has proven beneficial for both melody segmentation and melody retrieval. For instance, combining more than one melodic feature at a single $\mathcal{F}_1$ field is unfavourable because pitch sequence categories and rhythm categories can not be recognized independently of each other at $\mathcal{F}_2$. Indeed, the design of the event field ($\mathcal{F}_0$) facilitated this parallel and distributed representation of a melody. Furthermore, the separation of distinct features also yields a better compression of each learned melody. This is because each individual pitch sequence and rhythm only need to be learned and stored once in LTM. For example, the same rhythm can be associated with many pitch sequences (and vice versa). This would not be the case if more than one feature was represented at a single SONNET input field.

14.2.2 Improvements to SONNET

With the advances made in this dissertation regarding architecture, representation and application to melody segmentation and retrieval, a number of limitations and problems with previous SONNET formulations were discovered. To solve these problems, the following modifications were introduced:

- New work on presynaptic inhibition showed how multiple cell assembly representations allowed repeated phrases to be adequately processed, even if the repeated phrases overlap. This modification proved vital for the segmentation work (described in Chapter 12) because cell assemblies can better track, and therefore recognize, their patterns when they are embedded within a large input stream. This is the first time that a SONNET implementation has been capable of handling phrase repetitions properly.
- A new multiplicative confidence formulation, which adequately measures arbitrary levels of mismatch between STM and LTM was introduced. This formulation is not introduced as a replacement to Roberts additive confidence formulation, but merely an alternative to be used depending on the requirements a particular application demands.

14.2.3 Introduction of SONNET-MAP

The organization of SONNET modules in a parallel fashion has undergone little or no investigation. To redress this balance, Chapter 10 introduced the SONNET-MAP neural network, which is configured in a parallel fashion. Each parallel SONNET module can represent a distinct feature of a multidimensional, temporal input sequence; in this case, a melody. In this thesis, the first module processes pitch sequences, while the second module processes rhythms. An additional hierarchical module may be used to aggregate classifications of pitch sequences and their corresponding rhythms in order to form a parallel and hierarchical organization of memory. However, a variety of practical, technical and operational problems arose when attempting to achieve such a parallel and hierarchical configuration of SONNET modules. These problems and their solutions are summarized next.

Consistency of working memory. Since multiple input fields are used to represent input sequences, with each field representing a distinct input dimension, the entire collection of input fields constitutes SONNET-MAP's working memory. In this thesis, a melody is represented by the dimensions of pitch ($\mathcal{F}_1^a$ and $\mathcal{F}_2^a$) and rhythm ($\mathcal{F}_1^b$ and $\mathcal{F}_2^b$). Since two input assemblies activate in response to each note onset, mechanisms are required to ensure the coordinated reset of these input assemblies. Otherwise, due to input field overloads and the chunking-out process, the short-term memory trace for the melody will become inconsistent across the entire working memory. This problem was addressed by allowing the overload reset mechanisms to operate independently at first. Then, when each module has had a sufficient time to learn patterns at their respective dimension, the chunking out process will maintain consistency of working memory. This is achived by using the new methods of boundary point alignment and classification synchronization.

Boundary point alignment — forming consistent boundary points. Unless a set of well defined interactions are specified between the input fields ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) and the classification fields ($\mathcal{F}_2^a$ and $\mathcal{F}_2^b$) of each parallel SONNET module, the categories used to perform a segmentation at separate SONNET modules may conflict. This is because it is highly unlikely that every phrase formed at these modules will be aligned along the same boundary points. This is due to the fact that different grouping cues at $\mathcal{F}_1^a$ (pitch-based) and $\mathcal{F}_1^b$ (rhythm-based) determine the boundary points at those modules. Unless this issue is addressed, the overlapping nature of the learned categories at each module will hinder the formation of multidimensional phrase representations. This prevents the maintenance of a consistent working memory. To resolve this issue, a method called boundary point alignment was introduced. Boundary point alignment occurs only during the segmentation phase when when a cell assembly is chunking-out its pattern. This process ensures that every dimension of a melodic phrase is learned when separate modules are used to process different dimensions of the phrase. This was achieved using associative maps at $\mathcal{F}_1$ and an orienting subsystem which monitors the chunking-out process at $\mathcal{F}_2$.

Classification synchronization — forming multidimensional phrase representations. Subsequent to the the segmentation phase and boundary point alignment, there is no guarantee of the synchronous chunking-out of the categories formed at boundary points. This is because the parallel modules would be still operating in isolation from one another in terms of actually performing the chunking-out of a pattern. As a consequence, the possibility of contradictory chunk-outs subsequent to the segmentation phase may lead to an inconsistent working memory and its related problems. Furthermore, unless multidimensional phrase representations are formed, multidimensional phrase hierarchies can not form. Therefore, a method of binding categories that represent different dimensions of the same melodic phrase is required. Classification synchronization ensures that every dimension of a multidimensional phrase is classified and chunked-out at the same time. Associative maps are used to *bind* the cell assemblies that represent the different dimensions of a pattern together. In this case, pitch sequence representations at $\mathcal{F}_2^a$ are bound to their corresponding rhythm representations at $\mathcal{F}_2^b$. Then, the neural pathways that exist between each dimension may be used to perform associative priming. Priming signals can then be used to ensure that chunking-out is synchronized across all dimensions.

Forming a hierarchical phrase structure. Once the problems discussed so far have been addressed, a multidimensional phrase hierarchy can be formed. However, unlike the SONNET hierarchy described in Chapter 7, input to a single $\mathcal{F}_3$ classification field will originate from two different sources; a pitch sequence classification and a rhythm classification. Although only minor modifications are required, this is the first time that such a structure has been specified using SONNET-based networks. (see Section 10.7).

Optimization of resources. If SONNET-MAP is to be scaled-up to very large input sets, the optimization of resources becomes important. Section 10.8 shows how SONNET-MAP can be empowered to dynamically determine its own size depending on how learning progresses.

As a consequence of SONNET-MAP's architecture and the techniques listed above, this is the first time a SONNET-based neural network has been capable of all of the following:

- Automatically segmenting a multi-dimensional temporal input stimulus. This allows more than one dimension at a time to influence the segmentation process.
- Associating (or binding) phrases that represent different dimensions of the same temporal input stimulus.
- Forming a two-dimensional phrase hierarchy.

This parallel and hierarchical configuration has a number of advantageous over single dimensional models. In addition to improving recognition and yielding a greater compression of learned sequences, the use of parallel modules allows multiple dimensions to be utilized during category formation, which generally improves the segmentations formed. Furthermore, this architecture provides a framework that may be used to investigate how other temporal patterns may be associated. For example, other cognitive tasks like sign language and speech recognition, or even applications outside cognition such as weather forecasting or stock market prediction (see Section 14.3.4).

In this thesis, SONNET-MAP was solely used for segmenting melodies, utilizing the perceptual grouping cues of pitch and rhythm to enhance the segmentation process. The boundary points formed using SONNET-MAP were then used to populate the *Re*TREE*ve* network. More on these segmentation results will be discussed in Section 14.2.5.

14.2.4 Introduction of *Re*TREE*ve*

Although SONNET hierarchies and SONNET-MAP can be used effectively to segment melodies, a number of issues arise when the these networks are applied to the problem of the CBR of melodies. These problems prevent efficient, accurate and robust melody recognition and retrieval:

- Previous SONNET-based research has dealt with small data sets [114, 118, 133]. These data sets served their purpose in illustrating SONNET's application to the task in question. However, when the task at hand is CBR, large data sets are common. Therefore, scalability and performance considerations need to be taken into account.
- So far, SONNET-based research has focussed on the problems of segmentation, learning and recall. The desired behaviour was achieved using winner-take-all (WTA) competitive fields at $\mathcal{F}_2$. However, when CBR is desired, global constraints need to to be satisfied too. Section 11.2.3 shows that WTA competition is too restrictive for CBR.

Consequently, the *Re*TREE*ve* architecture and CBR algorithm was invented to address these issues. *Re*TREE*ve*'s architecture and operation is markedly simplified with respect to SONNET-MAP. With *Re*TREE*ve*, pre-segmented melodies can be efficiently used to populate the network. Then, retrieval is performed using a match-based retrieval algorithm that processes multiple winners at $\mathcal{F}_2$. This approach has a number of advantages:

- A melody is stored as a learned hierarchy of automatically segmented melodic phrases (with the pitch and rhythm dimensions stored independently) rather than as a string of symbols. In effect, this approach narrows down the search space because matches occur at the level of entire phrases rather than individual notes. Moreover, since the segmentation process is inspired by human perception and memory, queries are more likely to be aligned along the boundary points specified by these phrases and, as the empirical results show, queries that coincide with boundary points are generally retrieved using fewer notes.

- *Re*TREE*ve* allows parallel modules to influence each other, which enhances the retrieval process, particularly when one dimension of a query is poorly specified with respect to the other.

- Due to the manner in which segmented pitch sequences and rhythms are populated and stored independently, a natural compression of the amount of information stored in *Re*TREE*ve* takes place. This is because only a single copy of each pitch sequence and each rhythm needs to be stored. In music, pitch sequences often recur using different rhythms and vice versa.

- The *direct access* property of *Re*TREE*ve* means that retrieval speed is constant. In other words, retrieval times are independent of where a melody is actually located, albeit this point is only valid for systems implemented on parallel computers.

14.2.5 Empirical findings

Segmentation

As far as we are aware, SONNET-MAP is the first multidimensional neural network that is capable of automatically segmenting melodies based on the perceptual grouping cues of pitch sequences and rhythms. In Chapter 12, SONNET-MAP was applied to the task of segmenting the catalogue of *Beatles* melodies listed in Appendix E. These results showed that SONNET-MAP did indeed work as intended and, in particular, the following observations were observed:

- Both pitch and rhythm influenced the segmentation process. This is particularly desirable because, as pointed out by Page [118, pg. 240] with respect to a SONNET hierarchy, the use of additional grouping cues may help prevent the chunking-out of *"across-phrase sequences"*.

- The dimension of time dominated the segmentation process, which is consistent with human melody perception.

- The segmentations were consistent with various other aspects of human melody perception; such as pitch proximity, temporal proximity, pattern repetition, the gap principle and the primacy effect.

To date, the empirical tests presented in Chapter 12 represents the largest ever test of SONNET-based research.

Retrieval

The *Re*TREE*ve* network was trained using the boundary points discovered by SONNET-MAP's segmentation of *The Beatles* melodies. Then, *perfect queries*, *noisy queries* and *noisy queries with insertions and deletions* were presented to evaluate its retrieval performance. The empirical results show that:

- Queries that are aligned along phrase boundaries are retrieved more effectively that queries that are not aligned along phrase boundaries. That is, fewer notes are required to achieve a correct retrieval. This indicates the benefits of melody segmentation, and in particular, pursuing human-based segmentation techniques.
- The use of a multidimensional hierarchy significantly outperforms the use of a single-dimensional hierarchy.
- Retrieval performance degrades gracefully as the quality of the queries degrade. Hence, recognition performance is robust.
- Particular melody representations are more suited to particular query types. For example, a pitch-interval representation is better suited to very noisy queries than the IPC representation. Conversely, the IPC representation is more suited to perfect queries than pitch-interval and melodic contour representations.

14.3 Further research

14.3.1 SONNET-based research

The inspiration behind the use of presynaptic inhibition to process repetitions of items and phrases stems form Nigrin's theoretical specification of SONNET 2 [114], which solely uses presynaptic inhibition for all forms of competition. We believe this has the potential to be a powerful idea in ANN theory [92, 153]; consequently further research on a possible SONNET 2 implementation should be investigated. Furthermore, Section 11.2.2 pointed out the performance problems of implementing an inherently parallel architecture for processing temporal sequences on a serial computer. We believe this situation can be vastly improved by utilizing multiprocessor hardware for implementing SONNET-based neural networks. This would allow these networks to operate on a much larger scale and to be applied to a wider variety of applications. For example, SONNET-MAP (and indeed *Re*TREE*ve*) could be extended beyond two dimensions. This would allow many more features of an input stimulus to influence processing. Additionally, SONNET-MAP could also be extended hierarchically to allow the formation of more abstract categories.

14.3.2 Melody segmentation and retrieval

The application of an ANN approach to melody segmentation and retrieval is relatively new to the field of MIR. Although this thesis introduces such an approach, we believe there is great potential for further research. Melody segmentation performance (and indeed retrieval) can be improved further by utilizing as many features as possible during the segmentation process by extending SONNET-MAP beyond two dimensions. For example, aspects of tonality such as cadence points could be utilized in the formation of boundary points. Furthermore, the use of the IPC representation could be enhanced by building in pitch similarity measures, such as those of the circle of fifths or Shepherds pitch helix [147]. This would be particularly beneficial for melody retrieval and is one area we strongly recommend for further research.

Although the empirical results presented in thesis represent the largest test of a SONNET-based neural network reported to date, commercial retrieval systems will generally operate on a much vaster set of data. Therefore, SONNET-MAP and *Re*TREE*ve* should be tested on a much larger scale. Furthermore, one weakness regarding melody retrieval in general is that of comparing the performance of different retrieval systems. Each research project tends to operate on its own set of input data and uses its own set of test queries, which are derived in various different ways. In other words, there is no standard collection of music, queries or evaluation metrics to compare retrieval performance on a level playing field. This makes genuine comparisons between different retrieval systems extremely difficult. Nevertheless, Downie [41] and Downie et al. [43, 44] have recently introduced a framework to address these issues. In light of this, the future evaluation of SONNET-MAP and *Re*TREE*ve* should and must be performed within this new framework.

14.3.3 Music processing in general

Due to the ability of SONNET-MAP to form a hierarchy, higher level global melodic structures, such as verse and chorus repetitions, could be inferred from a melody. This would be particularly useful in applying SONNET-MAP to the problem of melody composition (see Mozer [107] and Todd [159] for more information). With respect to melody recall, it would be interesting to extend Page's expectancy work to include interval-based encoding schemes. This would enable SONNET-MAP to recall all possible features of a melody, regardless of what representational schemes are utilized at $\mathcal{F}_1$. Finally, although all SONNET research to date has focussed on monophonic music, the possibility of extending SONNET to process polyphonic music would be worth investigating.

14.3.4 Other fields of research

Although almost all SONNET research has focussed on processing musical sequences, we believe that SONNET could be utilized effectively in other research fields such as language, vision and time-series prediction. For example, a SONNET-MAP network could be used as a sign language translator. One SONNET module could process the sequences of hand and arm gestures inherent in sign language, while another associated SONNET module could translate these gestures into text or speech. Conversely, one module could process speech while another could translate the speech into sequences of hand gestures on a video screen. Furthermore, the potential of SONNET hierarchies to discover sequential regularities over long time periods could be utilized to discover sequences of patterns which lead to particular conditions or events occurring. This may be useful in predicting changes on the the stock market or predicting the weather.

Part V

Appendices

Appendix A

SONNET Specification

A.1 Introduction

This appendix specifies the SONNET neural network, encompassing the work presented in Chapter 5, Chapter 6 and the modifications introduced in Chapter 8. Additionally, some of the modifications presented in Section 7.3 are also included. That is, the T_{ni}, τ_{ni}, z^{-}_{mjni} and link interaction modifications. This is a generic specification of SONNET, which can be configured to operate in either event-driven (ED) mode or time-driven (TD) mode. Furthermore, the network will behave differently depending on its architectural configuration; that is, whether a SSG architecture or a MSG architecture is utilized.

A.2 Network architecture and indexing scheme

A SONNET module consists of two fields of assemblies: $\mathcal{F}_1$ and $\mathcal{F}_2$. These fields are organized as follows:

- $\mathcal{F}_1$ contains N_1^g sibling groups. Each sibling group contains N_1^s input assemblies (or siblings). The m^{th} sibling in the j^{th} sibling group is labelled γ_{mj}. Each input assembly consists of an input cell (s_{mj}) and a feedback cell (f_{mj}).
- $\mathcal{F}_2$ contains N_2^g sibling groups. Each sibling group contains N_2^s cell assemblies (or siblings). The n^{th} sibling in the i^{th} sibling group is labelled χ_{ni}. Each cell assembly consists of an excitatory cell (b_{ni}), a classification cell (c_{ni}), a feedback cell (d_{ni}) and an inhibitory cell (e_{ni}).
- Each $\gamma_{mj} - \chi_{ni}$ pair is connected by N_l interacting bottom-up links. Each link is indexed by k, where $k = 1$ indicates the strongest link and $k = N_l$ indicates the weakest link. The activity of the input assembly that traverses the k^{th} strongest link from the j^{th} sibling group to χ_{ni} is derived from the activity of γ_{kjni}, where γ_{kjni} is equal to γ_{mj} if:

$$R_{mj} = (k-1) + n_{jni} \tag{A.1}$$

where R_{mj} is the rank number of γ_{mj}, representing the number of other input assemblies in the j^{th} sibling group with larger input cell activations. Note that, in the case where input assemblies have identical activation levels (when $s_{mj} = 0$), there will be a tie. In this circumstance, the tie is broken arbitrarily. n_{jni} is the number of $\mathcal{F}_1$ input assemblies in the j^{th} sibling group with shut-off links to χ_{ni}. Together, the quantities R_{mj} and n_{jni} are used to process item and phrase repetitions (see Sections 6.2 and 8.2 respectively).

- All cell assemblies at $\mathcal{F}_2$ are connected to one another by inhibitory connections, with the exception of cell assemblies that reside in the same sibling group. The inhibitory connection from the m^{th} cell assembly in the j^{th} sibling group (χ_{mj}) to the n^{th} cell assembly in the i^{th} sibling group (χ_{ni}) is labelled z^{-}_{mjni}.

Depending on the architectural configuration of SONNET, its behaviour will differ slightly. If multiple sibling groups are specified at $\mathcal{F}_1$ (that is, $N_1^g > 1$), K_{msg} = true, otherwise K_{msg} = false.

A.3 Initializing the network

All of the network quantities are initialized to zero, except for the following:

- The quantities D_{ni} and I^{max}_{ni} are set to unity.
- Each excitatory bottom-up weight, z^{+}_{kjni}, is set randomly within the range $[K_{zmin}, K_{zmax}]$. This range must be set in a manner that remains consistent with the minimum bottom-up weight vector length, z_{min}. Additionally, after initialization, the range must not be set such that $z^{tot}_{ni} > K_5$. This would imply that a partially encoded pattern has been learned, which is impossible since no learning will have occurred yet. Finally, since the initial bottom-up input to $\mathcal{F}_2$ is influenced by the excitatory weights, K_{zmin} and K_{zmax} should be set so as to prevent extremely slow initial learning.
- Each secondary weight, w^{+}_{kjni}, is set to K_{winit}.
- Initially, every sibling group at $\mathcal{F}_2$ may only have one active cell assembly. That is, a single cell assembly with χ^{active}_{ni} defined as *true*. Subsequently, sibling groups that become committed to a pattern may have more than one sibling active at any given time. As explained in Section 8.2, this enables phrase repetitions to be processed.

A.4 Network equations

A.4.1 Preliminary issues

Event-driven mode. In event-driven mode ($K_{mode} = ED$), immediately after an event occurs, the network repeatedly adapts its state for a specified time period of T_{attn}. During this time period, each network quantity is calculated at points in time that are separated by the interval T_{step}. At these points in time the differential equations are integrated using the *Laplace* method, which is described in Section D.5. When the time period of T_{attn} has elapsed, the network becomes inactive until another event occurs.

Time-driven mode. For time-driven configurations of SONNET ($K_{mode} = TD$), the network continuously adapts its cells' activities and weights, regardless of whether an event has occurred or not.

Commit process. A cell assembly, χ_{ni}, becomes committed when $M_{ni} \geq K_M$ and $I^{+}_{ni} \geq 1$ for the first time. In this circumstance χ^{com}_{ni} is defined as true. Before this, however, χ^{com}_{ni} = false.

Multiplexing scheme. Each cell within a cell assembly at $\mathcal{F}_2$ outputs two quantities: the cell's activation level and the cell assembly's confidence value, $I^{\times}_{ni}$.

Parameter settings. The network's parameters are set in accordance with the guidelines presented throughout this thesis. In particular, see Section 12.2.4.

A.4.2 $\mathcal{F}_1$ dynamics

Input sequences are presented to SONNET one event at a time. In general, when an input event, e_i, occurs:

1. Every input cell is scaled using the following equation:

$$s_{mj}(i) = s_{mj}(i-1)\omega^{\lambda} \tag{A.2}$$

2. Any inactive input assembly within e_i's corresponding sibling group is set to μ.

where λ is a quantity that depends on the input representation being used (see Chapter 9 for more detail).

In event-driven mode, when any input assembly has been active for a time period of K_{12}, the three input assemblies that represent the oldest input events are reset. Additionally, the input assembly that represents the fourth-oldest input event may also be reset. There is a 50% chance of this happening. In time-driven mode, when an input assembly has been active for a time period of K_{12} it is reset. In event-driven mode K_{12} is expressed in terms of T_{attn}, whereas in time-driven mode K_{12} is expressed in terms of T_{span}

A.4.3 Excitatory bottom-up input to $\mathcal{F}_2$

The bottom-up input to each excitatory cell (b_{ni}) is calculated as follows:

$$b_{ni} = I^+_{ni} I^\times_{ni} \tag{A.3}$$

$$I^+_{ni} = \sum_{j=1}^{N^g_1} \sum_{k=1}^{N_l} S_{kjni} z^+_{kjni} \tag{A.4}$$

$$I^\times_{ni} = \left[K_\times + \left(\frac{I^*_{ni}}{|T_{ni}|} \right)^2 \right]^{I^*_{ni}} \tag{A.5}$$

$$I^*_{ni} = \sum_{k,j \in T_{ni}} I^\times_{kjni} \tag{A.6}$$

$$I^\times_{kjni} = \min \left(\frac{Z_{kjni}}{S_{kjni}}, \frac{S_{kjni}}{Z_{kjni}} \right) \tag{A.7}$$

$$Z_{kjni} = \frac{z^+_{kjni}}{z^{tot}_{ni}} \tag{A.8}$$

$$S_{kjni} = \begin{cases} \frac{s_{kjni}}{\left[\sum_{k,j \in T_{ni}} (s_{kjni})^2 \right]^{1/2}} & \text{if } k, j \in T_{ni} \\ \frac{s_{kjni}}{\left[\sum_{j=1}^{N^g_1} \sum_{k=1}^{N_l} (s_{kjni})^2 \right]^{1/2}} & \text{otherwise} \end{cases} \tag{A.9}$$

A.4.4 Learning excitatory bottom-up weights

The excitatory bottom-up weight (z^+_{kjni}) on the k^{th} strongest link from the j^{th} sibling group to b_{ni} is not updated when $\chi^{com}_{ni} = true$. Otherwise, it is updated as follows:

$$\frac{\partial}{\partial t} z^+_{kjni} = \epsilon_1 r_{kjni} c_{ni} \left[-L_{ni} z^+_{kjni} + S_{kjni} c_{ni} \right] \tag{A.10}$$

$$L_{ni} = \begin{cases} 1 & \text{if } z^{tot}_{ni} > z_{min} \\ 0 & \text{otherwise} \end{cases} \tag{A.11}$$

$$z^{tot}_{ni} = \left[\sum_{j=1}^{N^g_1} \sum_{k=1}^{N_l} (z^+_{kjni})^2 \right]^{\frac{1}{2}} \tag{A.12}$$

When $z^{tot}_{ni} > K_{embed}$, χ_{ni} can recognize embedded patterns, or even the beginning of these patterns (see Sections 6.2 and 7.3.9 for more detail).

The learning rate modulator (r_{kjni}) is calculated as follows:

$$r_{kjni} = \left[(T_{kjni})^{(1-T_{kjni})M_{ni}}(1-T_{kjni})^{T_{kjni}(1-M_{ni})}\right]^{f_{ni}} \tag{A.13}$$

$$T_{kjni} = 0.1 + \frac{0.8}{1+\exp\left[-4f_{ni}\left(\frac{w_{kjni}}{w_{0ni}} - \frac{1}{K_3}\right)\right]} \tag{A.14}$$

$$f_{ni} = \begin{cases} 20\sqrt{z_{ni}^{tot} - K_5} & \text{if } z_{ni}^{tot} > K_5 \\ 0 & \text{otherwise} \end{cases} \tag{A.15}$$

$$M_{ni} = \min\left(\frac{I_{ni}^{max}}{I_{ni}^{\times}}, \frac{I_{ni}^{\times}}{I_{ni}^{max}}\right) \tag{A.16}$$

$$\frac{\partial}{\partial t} I_{ni}^{max} = \begin{cases} 0 & \text{if } \chi_{ni}^{com} \\ \epsilon_1\left[-I_{ni}^{max} + 2I_{ni}^{\times}\right] & \text{not } (\chi_{ni}^{com}) \text{ and } (I_{ni}^{max} < I_{ni}^{\times}) \\ -\epsilon_1 r_{0ni} c_{ni} I_{ni}^{max} & \text{not } (\chi_{ni}^{com}) \text{ and } (I_{ni}^{max} \geq I_{ni}^{\times}) \end{cases} \tag{A.17}$$

where r_{0ni} is calculated using the r_{kjni} formulation with $T_{ni} = 0.9$.

A.4.5 Specifying the set T_{ni}

The set T_{ni} is defined as follows:

$$k, j \in T_{ni} \iff K_3 w_{kjni}^{+} \geq w_{0ni}^{+} \tag{A.18}$$

where w_{0ni} is the largest secondary weight to χ_{ni}.

The secondary weights (w_{kjni}^{+}) are not adjusted when $\chi_{ni}^{com} = true$, otherwise they are adjusted as follows:

$$\frac{\partial}{\partial t} w_{kjni}^{+} = \begin{cases} \epsilon_1 r_{kjni} c_{ni}\left[-L_{ni} w_{kjni}^{+} + I_{jni}^{\times} c_{ni}\right] & \text{if } K_{msg} = \text{false} \\ \epsilon_1 r_{kjni} c_{ni}\left[-L_{ni} w_{kjni}^{+} + c_{ni}\right] & \text{if } K_4 S_{kjni} > S_{0ni} \\ \epsilon_1 r_{kjni} c_{ni}\left[-L_{ni} w_{kjni}^{+}\right] & \text{otherwise} \end{cases} \tag{A.19}$$

where S_{0ni} is the largest normalized s_{kjni} activity to χ_{ni}.

A.4.6 χ_{ni} dynamics

The activation of each classification cell is governed by:

$$\frac{\partial}{\partial t} c_{ni} = -Ac_{ni} + \left(\frac{B - c_{ni}}{D_{ni}}\right)\left(\frac{v_1 b_{ni} + e_{ni}}{\tau_{ni}}\right) - c_{ni} v_2 \frac{Q_{ni}}{D_{ni}} \tag{A.20}$$

τ_{ni} represents the time dependence function used to control the influence that relatively long IOIs have on learning. τ_{ni} is dependent on the absolute time (t_{ni}), in seconds, since c_{ni} last fired. $\tau_{ni} = 1.0$ if $z_{ni}^{tot} > K_{embed}$ or if $K_{mode} = ED$, otherwise:

$$\tau_{ni} = \begin{cases} 0.75 & \text{if } t_{ni} \leq 0.5 \\ 3t_{ni}^2 & \text{if } 0.5 < t_{ni} \leq 1.0 \\ 3.0 & \text{if } t_{ni} \geq 1.0 \end{cases} \tag{A.21}$$

The inhibition received at χ_{ni} is calculated as follows:

$$Q_{ni} = \sum_{m,j \neq n,i} e_{mj} z^{-}_{mjni} \tag{A.22}$$

A ceiling of $(K_7 e^{big})$ is placed on cell assemblies in the set Q. This set consists of uncommitted cell assemblies with $z^{tot}_{mj} \leq K_5$ or $M_{mj} > K_M$. e^{big} is calculated as follows:

$$e^{big} = \max_{m,j \in Q} (e_{mj}) \tag{A.23}$$

The feedback and inhibitory cell activations are computed as follows:

$$e_{ni} = b_{ni} d^2_{ni} \tag{A.24}$$

$$d_{ni} = \min(c_{ni}, 1.0) \tag{A.25}$$

c_{ni} is reset to zero whenever an abrupt drop in b_{ni} occurs. Additionally, c_{ni} is reset to zero whenever the following conditions are simultaneously met:

- At least q uncommitted cell assemblies, χ_{mj}, have $K_{10} e_{mj} > e_{ni}$
- At least q uncommitted cell assemblies, χ_{mk}, have $I^{\times}_{mk} > I^{\times}_{ni}$
- At least one uncommitted cell assembly, χ_{ml}, has $e_{ml} > e_{ni}$ *and* $M_{ml} > K_M$

A.4.7 Self-organizing $\mathcal{F}_2$ into a masking field

A cell assembly's size (or dilution parameter) is not updated when $\chi^{com}_{ni} = true$, otherwise it is updated as follows:

$$\frac{\partial}{\partial t} D_{ni} = \epsilon_1 \epsilon_2 r_{0ni} \max\left[c^2_{ni}, K_D\right]\left[-D_{ni} + I^{\times}_{ni}\right] \tag{A.26}$$

The inhibitory weight from χ_{mj} to χ_{ni} evolves as follows:

$$z^{-}_{mjni} = \begin{cases} \epsilon_{inh} c^2_{ni} \left[-z_{mjni} + \frac{I^{\times}_{mj}}{\rho_{ni}}\right] & \text{if } K_{msg} = \text{false} \\ \epsilon_1 \epsilon_2 c_{ni} \left(-z^{-}_{mjni} + 1\right) & \text{if } (\chi^{com}_{mj} \text{ and } \chi^{com}_{ni}) \text{ or } (\rho_{ni} \leq I^{\times}_{mj}) \\ \epsilon_1 \epsilon_2 c_{ni} \left(-z^{-}_{mjni} - 1\right) & \text{not } (\chi^{com}_{mj} \text{ and } \chi^{com}_{ni}) \text{ and } (\rho_{ni} > I^{\times}_{mj}) \\ & \text{and } (z^{-}_{mjni} > 0) \\ 0 & \text{otherwise} \end{cases} \tag{A.27}$$

$$\epsilon_{inh} = \epsilon_1 \epsilon_3 \min\left[(I^{+max}_{ni})^{-5}, K_{inh}\right] \tag{A.28}$$

$$\rho_{ni} = \rho I^{\times}_{ni} \tag{A.29}$$

If $K_{msg} = true$, z^{-}_{mjni} is set to and frozen at zero if χ_{mj} and χ_{ni} are both committed and $I^{\times}_{mj} \geq K_9 e^{max}_{mjni}$ whilst $I^{\times}_{ni} \approx (K_{\times} + 1)^{|T_{ni}|}$. e^{max}_{mjni} is the largest $I^{\times}_{mj}$ value received from the inhibitory link connecting χ_{mj} to χ_{ni} (since the time both χ_{mj} and χ_{ni} have become committed).

A.4.8 Chunking-out a classified pattern from $\mathcal{F}_1$

When $\chi_{ni}^{com} = true$, $M_{ni} \geq K_M$ and c_{ni} has been sustained above K_{13} for a time period of K_{14} (during network adaption), χ_{ni} chunks-out its pattern. This is achieved by sending direct reset signals to each input assembly in the set T_{ni} via the feedback connections. Note that the last item of a pattern is not reset if $K_{chunk} = false$. Also, the value of K_{14} is dependent on the mode of operation. If $K_{mode} = ED$, K_{14} is specified in terms of T_{attn}, whereas if $K_{mode} = TD$, K_{14} is specified in terms of T_{span}. When a cell assembly has performed a chunk-out, c_{ni} is set to zero and χ_{ni}^{active} is set to false. This remains true until it becomes activated again. Remember that a cell assembly may become active when another cell assembly (χ_{mi} within the same sibling group) has recognized its entire pattern across $\mathcal{F}_1$, and when an event occurs which is part of the the i^{th} sibling group's learned pattern. It can be determined that a cell assembly is responding to its entire pattern if $I_{ni}^{\times} \geq K_M$ (see Section 8.2 for more detail).

A.5 Summary

This appendix completely specified the SONNET neural network. This specification is used to comprise the SONNET-MAP network introduced in Chapter 10 and which is fully specified in Appendix B. Note, if one wishes to use Page's expectancy work (see Section 7.2) with this SONNET specification, the modified $W_{ni}^{\times}$ formulation presented in Section 8.3 must be used. This is due to the new multiplicative $I_{ni}^{\times}$ formulation.

Appendix B

SONNET-MAP Specification

B.1 Introduction

This appendix specifies the SONNET-MAP neural network introduced in Chapter 10. The equations that define the SONNET modules, which partly comprise SONNET-MAP, are specified in Appendix A.

B.2 Network architecture and indexing scheme

SONNET-MAP is composed of two parallel SONNET modules; one which represents pitch sequences and another which represents rhythms. An additional alphabetic index is used to distinguish the fields, cells and weights of these separate modules where ambiguities may arise. $\mathcal{F}_1^a$ and $\mathcal{F}_2^a$ represent the input field and the masking field of the pitch module, while $\mathcal{F}_1^b$ and $\mathcal{F}_2^b$ represent the input field and the masking field of the rhythm module. These input fields ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) receive information regarding event onsets from $\mathcal{F}_0$. Associative map fields at the note level ($\mathcal{F}_1^{ab}$ and $\mathcal{F}_1^{ba}$) are used to form temporary associations between input assemblies at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$. Additionally, associative map fields at the phrase level ($\mathcal{F}_2^{ab}$ and $\mathcal{F}_2^{ba}$) are used to form permanent associations between cell assemblies at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$.

- $\mathcal{F}_0$ is divided into a field containing 88 cells, each responding to a specific CPH spanning the entire range of usable pitches.

- See Section A.2 for the architectural specification of $\mathcal{F}_1^a$, $\mathcal{F}_2^a$, $\mathcal{F}_1^b$ and $\mathcal{F}_2^b$.

- $\mathcal{F}_1^{ab}$ consists of $N_1^{g,b}$ sibling groups. Each sibling group contains $N_1^{s,b}$ associative cells (or siblings). This mirrors the architecture of $\mathcal{F}_1^b$. The k^{th} sibling in the l^{th} sibling group is labelled x_{kl}. $\mathcal{F}_1^{ab}$ is connected to $\mathcal{F}_1^b$ by two fixed one-to-one links (in both directions) between each $x_{kl} - \gamma_{kl}$ pair. Each $\mathcal{F}_1^{ab}$ associative cell (x_{kl}) is connected to every $\mathcal{F}_1^a$ input assembly (γ_{mj}) by adjustable connections labelled u_{mjkl}. To facilitate communication between $\mathcal{F}_1^{ab}$ and the input fields $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$, each input assembly contains two additional cells; a latch cell labelled l and an associative prime cell labelled a. The reciprocal map field, $\mathcal{F}_1^{ba}$, is almost identical in structure, mirroring $\mathcal{F}_1^a$.

- $\mathcal{F}_2^{ab}$ consists of $N_2^{g,b}$ sibling groups. Each sibling group contains $N_2^{s,b}$ associative cells (or siblings). This mirrors the architecture of $\mathcal{F}_2^b$. The p^{th} sibling in the q^{th} sibling group is labelled x_{pq}. $\mathcal{F}_2^{ab}$ is connected to $\mathcal{F}_2^b$ by two fixed one-to-one links (in both directions) between each $x_{pq} - \chi_{pq}$ pair. Each $\mathcal{F}_2^{ab}$ associative cell (x_{pq}) is connected to every $\mathcal{F}_2^a$ cell assembly (χ_{ni}) by adjustable connections labelled u_{nipq}. To facilitate communication between $\mathcal{F}_2^{ab}$ and the masking fields $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$, each cell assembly contains an additional cell; an associative prime cell labelled a. The reciprocal map field, $\mathcal{F}_2^{ba}$, is identical in structure, mirroring $\mathcal{F}_2^a$.

- The hierarchical SONNET module, which may be used if desired, consists of a standard SONNET masking field labelled $\mathcal{F}_3$. However, unlike standard SONNET modules, its working memory consists of the input layers of both $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$.

B.3 Initializing the network

All of the network quantities are initialized to zero, except for the following:

- See Section A.3 to see how to initialize $\mathcal{F}_1^a$, $\mathcal{F}_2^a$, $\mathcal{F}_1^b$ and $\mathcal{F}_2^b$.

- At $\mathcal{F}_1^{ab}$ and $\mathcal{F}_1^{ba}$, all the associative and fixed one-to-one connections are initialized to unity.

- At $\mathcal{F}_2^{ab}$ and $\mathcal{F}_2^{ba}$, the fixed one-to-one connections at are initialized to v_c, while the adjustable connections are initialized to K_{uinit}.

B.4 Network operation

B.4.1 Synchronizing the $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ overload schemes

The SONNET-MAP input fields ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) operate independently at first, each operating its own overload reset mechanism. After a specified number of epochs, when patterns have had a chance to form at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$, both overload reset mechanisms are disabled. At this point, input cells at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ will be reset solely on the basis of the chunking out process at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$. To ensure consistency at this point, any input cells that are older than the oldest input cell just chunked-out are also reset. Further consistency is ensured by the boundary point alignment and classification synchronization processes specified next.

B.4.2 Boundary point alignment

Each associative cell at $\mathcal{F}_1^{ab}$ is governed by the following activation equation:

$$x_{kl} = \begin{cases} 1 & \text{if } a_{kl}^b + I_{kl}^{+_{ab}} + G_1 \geq 2 \\ 0 & \textit{otherwise} \end{cases} \tag{B.1}$$

$$I_{kl}^{+_{ab}} = \sum_{j=1}^{N_1^{g,a}} \sum_{m=1}^{N_1^{s,a}} a_{mj}^a u_{mjkl} \tag{B.2}$$

$$G_1 = \begin{cases} 0 & \text{if } \mathcal{F}_1^a \text{ and } \mathcal{F}_1^b \text{ are both active} \\ 1 & \textit{otherwise} \end{cases} \tag{B.3}$$

where a_{mj}^a and a_{kl}^b activate in response to an event onset at $\mathcal{F}_0$ or a chunk-out at either $\mathcal{F}_2^a$ or $\mathcal{F}_2^b$ respectively. Note that $\mathcal{F}_1^{ba}$ operates in exactly the same way.

The following learning rule enables associations to form between input assemblies (at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) that activate in response to the same event onset at $\mathcal{F}_0$:

$$\frac{\delta}{\delta t} u_{mjkl} = a_{mj}^a u_{mjkl}(x_{kl} - u_{mjkl}) \tag{B.4}$$

Note that these associations only last as long as an input cell is active, after which the associations formed are unlearned by setting u_{mjkl} back to its initial value of unity.

Due to these temporary associations, the set $\mathcal{A}^b$ can be formed at $\mathcal{F}_1^b$ when a cell assembly at $\mathcal{F}_2^a$ chunks-out its pattern:

$$\gamma_{kl} \in \mathcal{A}^b \iff a_{kl} = 1 \text{ and } l_{kl} = 0 \tag{B.5}$$

A search process is then initiated in response to a chunking-out event. The goal of this search process is to find the cell assembly, χ_{choice} at $\mathcal{F}_2^b$, which most closely matches the pattern of onsets specified by the set $\mathcal{A}^b$. χ_{choice} is found using the following equation:

$$\chi_{choice} = \begin{cases} \max_{p,q \in \mathcal{M}} \left(z_{pq}^{tot}\right) & \text{if } \mathcal{M} \neq \emptyset \\ \min_{p,q \notin \mathcal{M}} \left(z_{pq}^{tot}\right) & \text{otherwise} \end{cases} \tag{B.6}$$

where,

$$\chi_{pq}^b \in \mathcal{M}^b \iff T_{pq}^b = \mathcal{A}^b \text{ and } M_{pq}^b \geq K_M \tag{B.7}$$

Once a cell assembly at $\mathcal{F}_2^b$ has been found (χ_{choice}), a fast-learn process performs the actual alignment as follows:

1. If $T_{choice} \neq \mathcal{A}^b$, then set $T_{choice} = \mathcal{A}^b$. This can be achieved by setting all of χ_{choice}'s secondary weights in the set $\mathcal{A}^b$ to unity, while setting every other secondary weight to zero. At this point, the normalized input vector S_{choice} should be re-computed.

2. The excitatory weight vector z_{choice}^+ should converge to the normalized input vector. This can be achieved by setting $z_{choice}^+ = S_{choice}$. At this point, $I_{choice}^\times$ should be re-computed.

3. D_{choice} should converge to $I_{choice}^\times$ (that is, $D_{choice} = I_{choice}^\times$).

4. Set $\chi_{choice}^{com} = true$ to indicate that learning is now complete at χ_{choice}.

When the alignment has been performed, χ_{choice} chunks-out its pattern. Chunking-out should occur whether or not χ_{choice} had previously learned its pattern or not. In this way, every dimension of a multidimensional phrase is learnt, thus forming consistent boundary points across the parallel SONNET modules. As a consequence of this chunking-out at $\mathcal{F}_2^a$ and χ_{choice}, associations can form between $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ due to the classification synchronization process which is specified next.

B.4.3 Classification synchronization

Each associative cell at $\mathcal{F}_2^{ab}$ (and $\mathcal{F}_2^{ba}$) is governed by the following activation equation:

$$x_{pq} = a_{pq}^b v_c + I_{pq}^{+_{ab}} \tag{B.8}$$

$$I_{pq}^{+_{ab}} = \sum_{i=1}^{N_2^{g,a}} \sum_{n=1}^{N_2^{s,a}} a_{ni}^a u_{nipq} \tag{B.9}$$

where a_{ni}^a and a_{pq}^b activate in response to chunk-outs at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ respectively.

The following learning rule enables associations to form between cell assemblies (at $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$) that chunked-out their patterns at approximately the same time:

$$u_{nipq} = \begin{cases} 1 & \text{if } (a_{ni}^a = 1) \text{ and } (x_{pq} \geq K_{althresh}) \text{ and } (u_{nipq}^{frozen} = false) \\ u_{nipq} & \text{otherwise} \end{cases} \tag{B.10}$$

Due to these associations, a chunk-out at one dimension may trigger the chunking-out at the other dimension. To achieve this, associative prime cells at each cell assembly activate as follows:

$$a_{pq}^b = \begin{cases} 1 & \text{if } (G_2 = 1) \text{ and } (x_{pq} = 1) \text{ and } (T_{pq}^b = \mathcal{A}^b \text{ and } M_{pq}^b \geq K_M) \\ 0 & \text{otherwise} \end{cases} \tag{B.11}$$

$$G_2 = \begin{cases} 0 & \text{if } \mathcal{F}_2^a \text{ and } \mathcal{F}_2^b \text{ are both active} \\ 1 & otherwise \end{cases} \tag{B.12}$$

Cell assemblies that are primed and respond to their complete pattern ($M_{pq} \geq K_M$) are not subject to any chunking-out delay and will, therefore, chunk-out their patterns immediately. In this way, the chunking-out of primed cell assemblies is facilitated.

B.5 Summary

This appendix specified the SONNET-MAP neural network introduced in Chapter 10. Appendix C will specify how the *Re*TREE*ve* architecture can be utilized as a CBR system for melodies.

Appendix C

*Re*TREE*ve* Specification

C.1 Introduction

This appendix specifies the *Re*TREE*ve* architecture and CBR algorithm introduced in Chapter 11.

C.2 Network architecture

The architectural configuration for *Re*TREE*ve* is specified below:

- $\mathcal{F}_0$ is divided into a field containing 88 cells labelled e_l, each responding to a specific CPH, spanning the usable pitch range of human hearing. Communication between $\mathcal{F}_0$ and each input field is achieved using the appropriate gating mechanism (see Chapter 9).
- Each input field ($\mathcal{F}_1^a$ and $\mathcal{F}_1^b$) consists of N_g sibling groups. Each sibling group contains N_s siblings labelled s_{mj}, where j indexes a specific sibling group and m indexes a specific sibling within that group. In addition to the activation of each cell, the event number of the note which caused the cell to activate is also recorded.
- Each match field ($\mathcal{F}_2^a$ and $\mathcal{F}_2^b$) consists of N_2 match nodes labelled m_i. Initially, every match node is considered to be connected to every input cell at its corresponding input field. Furthermore, the excitatory weight on each connection is initialized to zero.
- The template field at $\mathcal{F}_3$ consists of N_3 template nodes labelled t_k. Every template node is considered to be connected to every match cell at both $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$. Each connection has a fixed weight of unity.

C.3 Long-term memory structures

The manner by which the *Re*TREE*ve* network is populated with melodies is described in Section 11.4. As a consequence of this learning phase, *Re*TREE*ve*'s LTM contains three data structures (analogous to those of SONNET-MAP's) which are utilized during retrieval.

The learned weights at $\mathcal{F}_2$. Each match node, m_i, contains a set T_i, which consists of the set of indices that identify the specific sibling groups at $\mathcal{F}_1$ that constitute m_i's learned pattern. Each index in T_i has a corresponding excitatory weight, z_{ji}^+, which encodes the content of that pattern. T_i is specified in descending order of excitatory weight strength. Phrase tracking is achieved by ensuring that T_i always responds to the $|T_i|$ most recent note onsets occurring at its corresponding input field.

The learned associations at $\mathcal{F}_2$. Each match node, m_i, contains a set of indices specified by $\mathcal{A}_i$, which identify the match nodes, m_j, with which they are associated with. That is, each pitch sequence has a list of associated rhythms and vice versa.

The learned melody templates at $\mathcal{F}_3$. The set $\mathcal{T}_k$ at each $\mathcal{F}_3$ template node, t_k, specifies all of the pitch-rhythm phrase pairings of a particular melody. That is, the set of index pairs (representing $\mathcal{F}_2^a$ and $\mathcal{F}_2^b$ match nodes) which encode the phrase sequence of a melody. In addition, the order in which these these pairings are expected to occur is also specified. The first member of each pair denotes the i^{th} pitch sequence represented by m_i at $\mathcal{F}_2^a$, while the second member denotes the j^{th} rhythm represented by m_j at $\mathcal{F}_2^b$.

C.4 Retrieving melodies

C.4.1 Input field dynamics

The activity of the input cells operate in exactly the same manner as described previously, with one exception. Every match node at $\mathcal{F}_2$ only responds to the $|T_i|$ most recent note onsets at $\mathcal{F}_1$. Furthermore, the maximum size of this set is five; $|T_i| = N_l = 5$. Therefore, only the five most recent events need to be stored at $\mathcal{F}_1$. Consequently, input cells which are older than the five most recently activated input cells are reset.

C.4.2 Phrase matching and multiple winners at $\mathcal{F}_2$

For the CBR of melodies, m_i is used to quantify the match between a node's LTM weight vector and its STM input vector. m_i is calculated (only once per note onset) as follows:

$$m_i = \frac{I_i^*}{|T_i|} \tag{C.1}$$

$$I_i^* = \sum_{j \in T_i} I_{mji}^{\times} \tag{C.2}$$

$$I_{mji}^{\times} = \min\left(\frac{Z_{mji}}{S_{mji}}, \frac{S_{mji}}{Z_{mji}}\right) \tag{C.3}$$

$$Z_{mji} = \frac{z_{mji}^{+}}{\left[\sum_{j \in T_i} (z_{mji}^{+})^2\right]^{1/2}} \tag{C.4}$$

$$S_{mji} = \frac{s_{mji}}{\left[\sum_{j \in T_i} (s_{mji})^2\right]^{1/2}} \tag{C.5}$$

Any cell assembly is considered to match its pattern when m_i reaches the match threshold specified by K_M. This formulation allows multiple winners at $\mathcal{F}_2$ to be processed by $\mathcal{F}_3$.

C.4.3 Specifying associated pitch-rhythm phrase pairs

The set $\mathcal{P}$ is used during retrieval to yield a match at each template node at $\mathcal{F}_3$. $\mathcal{P}$ contains a list of the pitch sequence-rhythm pairs that match the current input query at $\mathcal{F}_1$. The i^{th} node at $\mathcal{F}_2^a$ and the j^{th} node at $\mathcal{F}_2^b$ are considered to be a matched pair (and consequently members of $\mathcal{P}$) if the following conditions are met:

- m_i^a has a learned association with m_j^b, that is if $j \in A_i$ (or vice versa).
- If m_i^a and m_j^b both match their corresponding input patterns at $\mathcal{F}_1^a$ and $\mathcal{F}_1^b$ respectively. That is $m_i^a \geq K_M^a$ and $m_j^b \geq K_M^b$.
- If m_i^a and m_j^b are aligned at the event level. This can be determined if $|T_i^a| = |T_j^b|$.

Mathematically, a template node at $\mathcal{F}_3$ can determine if a pitch sequence-rhythm pair specified in its template $\mathcal{T}_k$ is member of $\mathcal{P}$:

$$\mathcal{T}_{k_{ij}} \in \mathcal{P} \iff (m_i^a \geq K_M^a) \text{ and } (m_j^b \geq K_M^b) \text{ and } (m_j^b \in A_i^a) \text{ and } (|T_i^a| = |T_j^b|) \tag{C.6}$$

In effect $\mathcal{T}_k$ deals with expected matched pairs, whereas $\mathcal{P}$ deals with actual matched pairs that currently exist, based on the current input query. Note also, associated with each matched pair is a set of indices that identify the event numbers of the notes which resulted in the recognition of the matched pitch-rhythm pair. This set is labelled $\mathcal{E}$.

C.4.4 Computing the ranked list at $\mathcal{F}_3$ using the retrieval algorithm

Each template node, t_k, at $\mathcal{F}_3$ matches the phrase pairs specified by $\mathcal{P}$ against its melody template $\mathcal{T}_k$. The length of the largest phrase sequence, $\mathcal{P}_{seq}$, consistent with $\mathcal{T}_k$ constitutes t_k's match value. Adjacent elements of the set $\mathcal{P}_{seq}$ must be aligned. A pair of adjacent elements are considered to be aligned if:

Algorithm C.1 The *Re*TREE*ve* algorithm.

```
procedure ComputeMatch( $\mathcal{T}$ )
    $match = 0$
    for $n = 1$ to $|\mathcal{T}|$ do
        $t_{pair} = \mathcal{T}[n]$
        $\mathcal{P}_{sub} = \mathcal{P}[t_{pair}]$
        for $m = 1$ to $|\mathcal{P}_{sub}|$ do
            $p = \mathcal{P}_{sub}[m]$
            $currentMatch = SequencePhrases(p,\ \mathcal{T}, n + 1)$
            if $currentMatch > match$ then
                $match = currentMatch$
            end if
        end for
    end for
    return $match$
end procedure

procedure SequencePhrases( $p$, $\mathcal{T}$, n )
    $pairMatch = |p.\mathcal{E}|$
    $seqMatch = 0$
    if $n \neq |\mathcal{T}|$ then
        $t_{pair} = \mathcal{T}[n]$
        $\mathcal{P}_{sub} = \mathcal{P}[t_{pair}]$
        for $m = 1$ to $|\mathcal{P}_{sub}|$ do
            $p_{next} = \mathcal{P}_{sub}[m]$
            if $AreContiguous(p, p_{next}) ==$ true then
                $currentMatch = SequencePhrases(p_{next},\ \mathcal{T}, n + 1)$
                if $currentMatch > seqMatch$ then
                    $seqMatch = currentMatch$
                end if
            end if
        end for
    end if
    return $pairMatch + seqMatch$
end procedure
```

1. They occur in the correct order with respect to $\mathcal{T}_k$.

2. The note onsets of each element are contiguous. This can be determined by examining the event numbers specified by each pitch-rhythm pair (that is, $\mathcal{E}$). A certain degree of overlap is allowed, which is specified by the parameter $K_{overlap}$.

The length of $\mathcal{P}_{seq}$ (representing the template node's match) is measured in terms of unique onset occurrences represented by each phrase pair in the set. It is these match values that are used to produce a ranked list. The match value for each template node at $\mathcal{F}_3$ is specified by Algorithm C.1.

C.5 Summary

This appendix specified the *Re*TREE*ve* algorithm for the CBR of melodies. Once Chapters 5 to 11 are understood, Appendices A to C provide all of the necessary information required to implement this CBR system. For additional information on how we implemented this system, see Appendix D.

Appendix D

Implementation Issues

D.1 Introduction

When implementing a complex piece of software, numerous issues arise; from high level object-oriented design decisions to subtle implementation nuances. This was particularly true when implementing and simulating the SONNET-MAP and *Re*TREE*ve* architectures introduced in this thesis. This appendix is not intended as a detailed description of the entire process, but instead to highlight some of the issues encountered, which other researchers may find helpful when implementing the SONNET-MAP and *Re*TREE*ve* architectures themselves. In particular, this appendix briefly describes the object-orientated design of the networks and various implementation decisions that were taken; such as optimization, locality considerations, numerical integration techniques, persistence and logging.

D.2 Object-oriented software design

The object-oriented software life cycle, from analysis to design to implementation, can be likened to producing a doctoral dissertation. From an initial analysis (see Chapters 1–4), a general concept gives rise to concrete system requirements. The design of a system to meet such requirements (see Chapters 5–11) gives rise to a detailed design, which usually includes an object model, class diagrams and interaction diagrams. Finally, the implementation phase involves coding the design using a particular programming language on a particular operating system (in this case Visual C++ on Windows XP). Then, to verify that the implementation satisfies the system requirements, testing is carried out (see Chapters 12 and 13).

The core concept behind object-orientation is that of an *object*. An object consists of a packet of *data* and a set of *methods* that are permitted to manipulate this data. Hence, methods characterize the behaviour of an object. Furthermore, since methods describe the communication interface between an object and its data, this data is effectively hidden (or *encapsulated*) behind this interface. An object is an *instance* of a *class*. A class can be considered as a template for creating objects. A class defines the data and the methods that may be used by class instances. An object's data is manipulated when it receives a *message*, which is a request to *invoke* a particular method. This causes processing to occur, which is defined by the behaviour of that method in the class specification. Such requests may be accompanied with additional data to be used during this processing.

A variety of features make the object-oriented approach towards software development attractive (see Eliëns [49] for more information on these topics):

Inheritance. Inheritance is a mechanism that enables the reuse and extension of class specifications through sub-typing (or specialization), which is achieved though behavioural refinement and polymorphism. This enables applications to be easily extended to incorporate additional functionality through incremental development and code reuse. Furthermore, the resulting tree structure (representing inheritance relationships and known as a *class hierarchy*) allows, and perhaps more importantly, design reuse.

Polymorphism. The ability to hide different implementations behind a common interface through polymorphism facilitates code extensibility and code reuse. This includes techniques such as overloading, templates (generic classes) and, as we have already mentioned, inheritance (dynamic binding).

Encapsulation. The ability to separate and change the internal workings of an object without changing other parts of the system, through the use of encapsulation, makes incremental development and code maintenance easier.

Modularity. The inherent modularity of the object-oriented approach enables developers to organize their implementations with relative ease. Furthermore, on a purely human level, the use of objects makes designing and understanding the design fairly intuitive.

The software design described here, based on the specifications of Appendices A through C, was designed with all of these issues in mind. The following list briefly describes the principal classes used in the SONNET-MAP neural network design (see Figure D.1):

`NetworkComponent`. This object defines a generic network component, from which, all network classes are derived.

`Hierarchy`. This object contains and governs the interactions between the modules and fields it contains. In the case of the SONNET-MAP hierarchy, modules for pitch, rhythm and phrase sequences are contained therein. Also, in addition to an event field, associative map fields at the event level and at the phrase level are also contained. See Figure D.2 for the containment relationships between all SONNET-MAP network components.

`Module`. This object contains and governs the interactions between an `InputField` and a `MaskingField`.

`Field`. A field contains assemblies organized according a particular topological specification. A field communicates with its containing module and its contained assemblies. Specialized fields

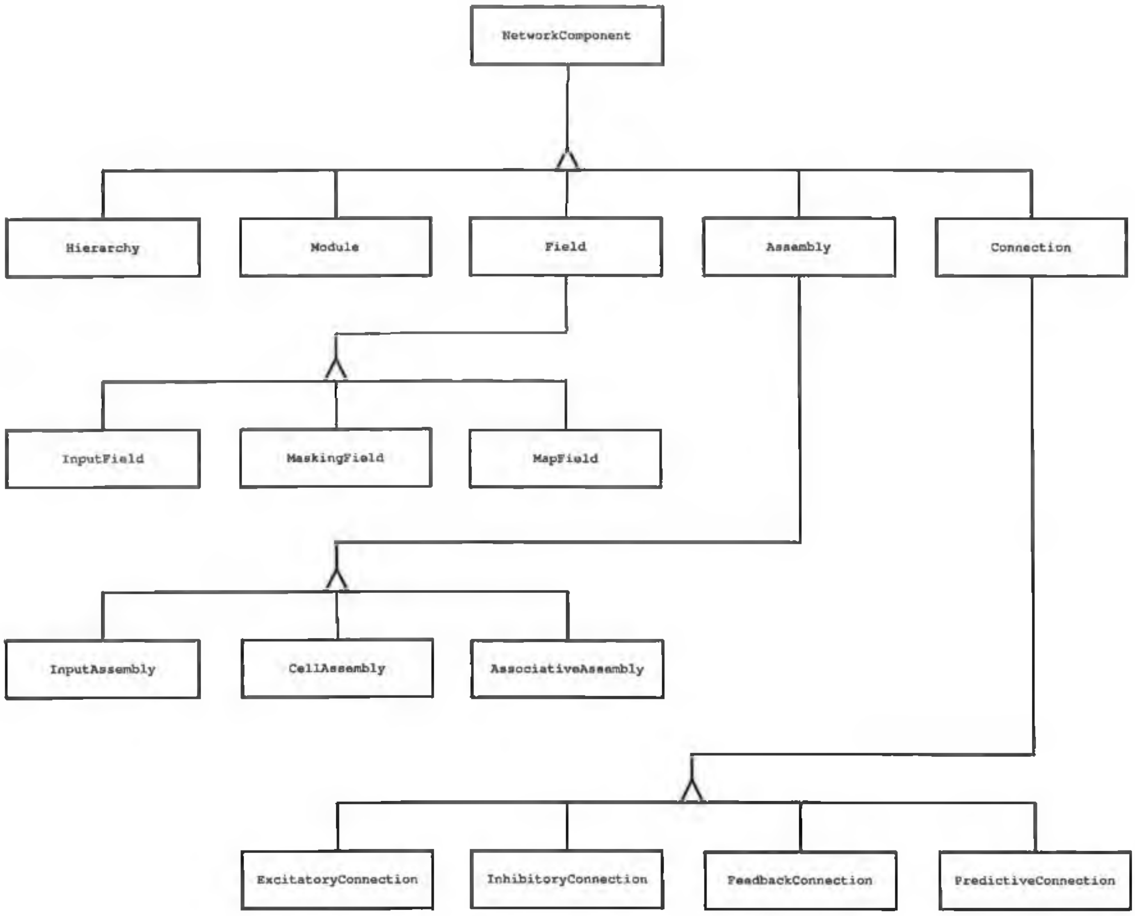


Figure D.1: Class diagram showing the core network components of the SONNET-MAP system design.

are used to contain the various types of assemblies. Namely, `InputField` ($\mathcal{F}_1$), `MaskingField` ($\mathcal{F}_2$) and `MapField` ($\mathcal{F}_2^{ab}$)

`Assembly.` Assemblies are one of the primary components of an ANN. An assembly may consist of a single cell or a group of cells. Their function is to calculate their own activation levels and/or adjust the weights on their incoming connections during learning. Assemblies can communicate via any connection that exists between them, or directly with their containing field. Typically, many assemblies are created. This software contains an `Assembly` base class, from which the following specializations are derived: `InputAssembly`, `CellAssembly`, and `AssociativeAssembly`.

`Connection.` Connections connect assemblies together. A connection has a particular weight, which evolves through a learning process. The input to a connection is the activation level of its source assembly or the external stimulus applied to it. It is convenient for connections to store references to its source and destination assemblies. Connections are organized according to a topological specification. The following specializations are derived from

Connection: `ExcitatoryConnection`, `InhibitoryConnection`, `FeedbackConnection` and `PredictiveConnection`.

The design described here contains base classes for ANN components that one would expect any ANN design to contain. The software was implemented using Visual C++ on Windows XP[1]. One point to highlight about this design is the use of the `NetworkComponent` class, from which all other classes are specializations. This makes life easier when adding new types of network components. For example, the basic components can be specialized in order to implement any specific ANN components. Although, using the design outlined here, ANNs ranging from multilayered feed-forward networks using back-propagation, recurrent networks using real-time recurrent learning, Hopfield networks and ART can be implemented, this thesis is concerned with implementing the SONNET-MAP neural network. This structure is common in object-oriented design and is neatly described as the *Composite* design pattern by Gamma et al. [54, pg. 163] — *"Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly."* .

Using these basic components, clients of this software can build ANNs according to any specified topology using any specified type of assembly. Likewise, they can then run the network on any training set using any specified learning algorithm. The only constraint is that the choices above are compatible. For example, it would not make sense to build a feed-forward network using Hopfield neurons. Since the user may specify any type of neuron to go with any particular network, the decision is made dynamically at runtime; therefore the network can not know ahead of time which type of neuron to create. For this reason, the creation of network components is performed via a *Builder* class, which is passed into the neural network constructor. Basically, the creation of a neural network is independent of the type of neurons that the network is to contain. Again, the idea of a Builder is described neatly by Gamma et al. [54, pg. 97] — *"Separate the construction of a complex object from its representation so that the same construction process can create different representations."*.

For more information on object-oriented design patterns and implementations using C++, refer to Gamma et al. [54], Meyers [102, 103].

D.3 Implementation optimizations

Section 11.2.2 considered the bottleneck encountered when implementing inherently parallel architectures on serial computers. A variety of mechanisms were introduced, which helped to resolve a variety of these issues with respect to segmenting and retrieving melodies. This section discusses a number of techniques that were utilized to generally reduce processing run-times. Many of these techniques exploit redundancies inherent in the SONNET-MAP network architecture.

[1] Page implemented his SONNET hierarchy using Objective-C on a NeXT workstation, while Roberts implemented his SONNET module using C++ on UNIX.

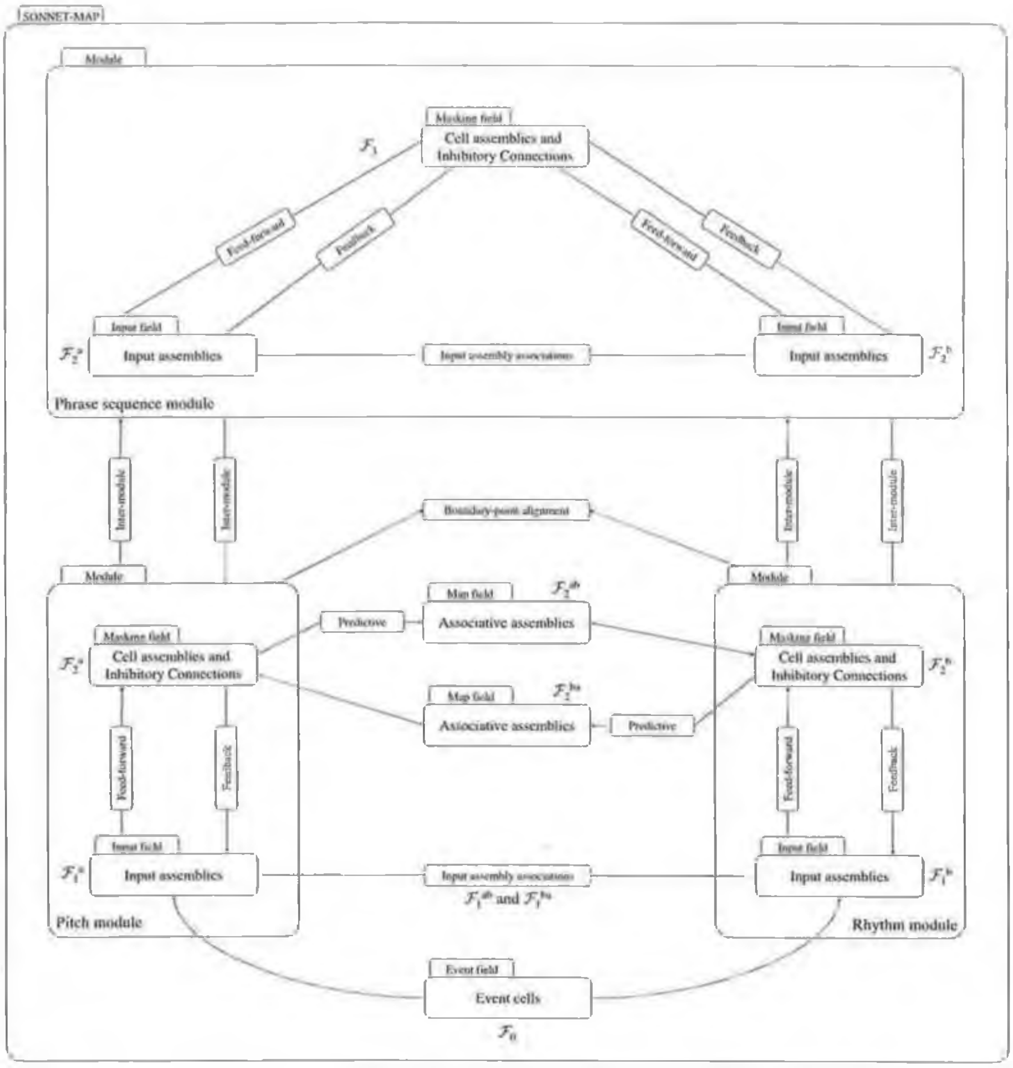


Figure D.2: Class containment diagram for the SONNET-MAP architecture.

D.3.1 Sharing bottom-up and top-down excitatory weights

Section 5.3.7 specified how the feedback weights, z^f_{nikj}, evolve and subsequently stabilize. That section also pointed out that the feedback weights evolve to become symmetrical to the feed-forward excitatory weights z^+_{kjni}. This redundancy enables a simple weight sharing mechanism to be employed. Since z^f_{nikj} becomes symmetrical to z^+_{kjni}, only a single set of weights need to be learnt, stabilized and stored. Therefore, the processing related to z^f_{nikj} (which is tedious, particularly the weight freezing mechanism) need not take place. Consequently the feedback weights simply reference the appropriate z^+_{kjni} quantities. For example, $\forall_{k,j,n,i}\ z^f_{nikj} = z^+_{kjni}$.

D.3.2 Sharing bottom-up LTM states

Section 6.2 described how repetitions at the event level can be processed. Additionally, inherent redundancies are utilized to efficiently model repetition handling. In effect, a weight sharing mechanism is utilized. Note that the connection and its associated LTM state is the finest level of granularity used for this and most other ANN implementations.

To implement the sharing scheme, a single connection from each input assembly at $\mathcal{F}_1$ is connected to each cell assembly at $\mathcal{F}_2$. Although only a single connection is utilized between each $\gamma - \chi$ pair, the LTM state may differ at that connection depending on the input assembly's rank and the number of links that are reset due to the relay-cell reset mechanism. Therefore, a pool of LTM states are referenced according to this information, which is shared among all connections. Gamma et al. [54, pg. 195] classifies this use of state-sharing in terms of the *Flyweight* design pattern — *"Use sharing to support large numbers of fine-grained objects efficiently."*. Furthermore, the separation of a state from its connection is know as an *envelope-letter* pair, where the connection represents the envelope and the state represents the letter.

D.3.3 Sharing χ_{ni} LTM states

Another redundancy at $\mathcal{F}_2$, arising from the requirement to handle phrase repetitions, can be exploited to reduce computation time. Section 8.2 showed that $\mathcal{F}_2$ should be organized in the same fashion as $\mathcal{F}_1$. In particular, each sibling within a particular sibling group at $\mathcal{F}_2$ should learn the same quantities listed below, which represent the LTM of each cell assembly:

- Excitatory weights (z^+).
- Cell size (D).
- Max confidence (I^{max}).
- χ^{com}.

Due to this redundancy, only a single version of each of these quantities needs to be learnt and stored at each $\mathcal{F}_2$ sibling group. This improves the implementation in terms of both space and time. This sharing scheme is easily implemented using a state pool, similar to that described in Section D.3.2, which all cell assemblies within the same sibling group can share. Note that the STM states (that is, the cell activations) cannot be shared in this way because they will be responding to different contexts (b, c, d, e, l and a).

D.3.4 Caching at χ_{ni}

Many network quantities only need to be updated in response to certain specific events. These events include:

- LTM changes at a cell assembly. That is, changes in χ_{ni}^{com}, D_{ni}, I_{ni}^{max} and z_{mjni}^{+}.
- The firing of an input cell. In this case, all cell assembly caches are invalidated.
- The resetting of an input assembly, either due to chunking-out or input field overload. In this situation, all cell assembly caches are invalidated.

The quantities that are updated in these circumstances include: I_{ni}^{+}, $I_{ni}^{\times}$, I_{ni}^{+max}, M_{ni}, ρ_{ni}, r_{ni} and L_{ni}. Consequently, rather than continuously updating these quantities, each quantity is stored in a cache, which may only be invalidated in response to the above events. This style of caching methodology is well known and is widespread in software implementations, and greatly reduces the running times of simulation trials.

D.4 Locality considerations

Nigrin went to great lengths to ensure that the locality property of biological neural networks was met by his SONNET specification. For example, the introduction of multiplexing (see Section 5.3.6). This consideration has been included in subsequent SONNET research. For example, Roberts I^{+max} introduction. In this thesis, for example, the boundary point alignment process and the use of attentional signals were inspired by the need to meet this property.

From a biological and specification point of view, we still consider this issue important. However, from an implementation point of view, particularly when ANNs are implemented on a serial computer, it would be naïve and impractical to implement neural architecture in this way because running time will tend to be very slow.

D.5 Numerical approaches to integration

Differential equations may be integrated using a variety of numerical methods. For example, *the Euler method*, *the Runge-Kutta method* or *the Laplace method*. In effect, the network simulations are performed in *pseudo real-time*. Page [118, Section 6.4] described using the fourth-order Runge-Kutta numerical method to integrate the differential equations relating to the change in cell activations. Furthermore, the first-order Euler algorithm was used to integrate the differential equations relating to the change in LTM weights. In each case, the step size was small enough to ensure the stability of the integration process. Roberts [133, Sectio 3.3.2] described using Laplace transforms to integrate all of the differential equations. From preliminary network simulation, Roberts found that (with $h = 2ms$) this method gave the same results as the Runge-Kutta method to four significant figures. In this thesis, the Laplace method described by Roberts was adopted.

All of the network equations can be written as follows:

$$\frac{\partial}{\partial t}x = a_1(-a_2x + a_3) \tag{D.1}$$

Now using Equation A.20, the problem becomes how to write:

$$\frac{\partial}{\partial t}c_{ni} = -Ac_{ni} + \left(\frac{B - c_{ni}}{D_{ni}}\right)\left(\frac{v_1b_{ni} + e_{ni}}{\tau_{ni}}\right) - c_{ni}v_2\frac{Q_{ni}}{D_{ni}} \tag{D.2}$$

in terms of:

$$\frac{\partial}{\partial t}c_{ni} = a_1(-a_2c_{ni} + a_3) \tag{D.3}$$

This is achieved as follows:

$$\frac{\partial}{\partial t}c_{ni} = -\left[A + \left(\frac{v_1b_{ni} + e_{ni}}{D_{ni}\tau_{ni}}\right) + \frac{v_2Q_{ni}}{D_{ni}}\right]c_{ni} + B\left(\frac{v_1b_{ni} + e_{ni}}{D_{ni}\tau_{ni}}\right) \tag{D.4}$$

Setting,

$$a_1 = 1.0 \tag{D.5}$$

$$a_2 = A + \left(\frac{v_1b_{ni} + e_{ni}}{D_{ni}\tau_{ni}}\right) + \frac{v_2Q_{ni}}{D_{ni}} \tag{D.6}$$

$$a_3 = B\left(\frac{v_1b_{ni} + e_{ni}}{D_{ni}\tau_{ni}}\right) \tag{D.7}$$

For further information on numerical integrations, see Boyce and DiPrima [10] and Edwards and Penney [47].

D.6 Persistence of learned quantities

After SONNET-MAP has learned a set of melodies, the learned values must be saved to a database if one wants to make further use of the network. The LTM values that are required to be saved include: z^+, z^f, D, I^{max} and flags such as those indicating whether a weight is frozen or not, or whether χ_{ni}^{com} is true or false.

D.7 Logging issues

To record the salient data required during a simulation run, in order to report on the segmentation and retrieval of melodies performed by SONNET-MAP and *Re*TREE*ve*, the use of a logging class is utilized. The logging class is used to record information such as the occurrence of a chunk-out, a commit or the extraction of a ranked list. A reference to this logging class is provided to SONNET-MAP's constructor. Then the relevant pieces of information can be recorded directly by the logging class. This enables the separation of logging functionality from the core ANN implementation.

D.8 Summary

This appendix briefly outlined some of the important issues dealt with while implementing the specifications laid out in Appendices A through C. Appendix E specifies the file structure of each melody and the entire catalogue of melodies used during the empirical testing described in Chapters 12 and 13.

Appendix E

Input Melodies used in Empirical Tests

E.1 Introduction

This appendix lists the entire set of fifty melodies used in the segmentation experiments presented in Chapter 12 and the CBR experiments described in Chapter 13. This catalogue's range spans the entire set of recordings made by *The Beatles*. Music notation is provided for ten of these melodies, which are used to illustrate how boundary points within a melody can be discovered using SONNET-MAP (see Chapter 12). The purpose of presenting the notation for these melodies is to enable other researchers to reproduce these results. The use of this test suite represents the largest and most difficult examination of a SONNET-based neural networks performed to date.

E.2 Structure of melody input files

A melody is presented at $\mathcal{F}_0$ as a sequence of {CPH, duration} duples. Rests and beats are special cases and are represented by the duples {*R, duration*} and {*B, duration*} respectively. A melody's tactus-span and key are also supplied. The durations are given in terms of the tactus-span, allowing the tempo of a melody to be changed by altering this single quantity rather than by altering each individual duration. Table E.1 illustrates the structure of the melody input file for *Mother Nature's Son.*

Tactus-span	0.72 seconds
Key	D major

CPH	**IOI**
A3	1.0
A3	1.0
B3	1.0
B3	1.0
B3	1.5
B3	0.5
A3	2.0
D4	1.0
D4	1.0
C♯4	1.0
D4	0.5
C♯4	0.5
B3	3.0
R	1.0
C♯4	1.0
B3	0.25
E3	0.5
A3	1.25
F♯3	1.0
E3	0.25
F♯3	0.5
A3	0.5
B3	0.25
A3	1.0
B3	0.5
D4	0.25
B3	0.5
D4	2.25
R	2.0

Table E.1: The structure of the input file for the melody *Mother Nature's Son.* See Section E.4.2 for the music notation of this melody.

E.3 Catalogue of melodies

Albums	**Melodies**
Please Please Me	Do You Want To Know A Secret?
With the Beatles	All My Loving, I Wanna Be Your Man
Beatles for Sale	No Reply, I'll Follow The Sun, Eight Days A Week
A Hard Day's Night	A Hard Day's Night, If I Fell, Can't Buy Me Love
Help!	Help!, Ticket To Ride, Yesterday*
Rubber Soul	Norwegian Wood(This Bird Has Flown), Michelle, In My Life
Revolver	Taxman, Eleanor Rigby, Here There And Everywhere, Yellow Submarine†
Sgt. Pepper's Lonely Hearts Club Band	When I'm Sixty-Four*, A Day In The Life
Yellow Submarine	Yellow Submarine†, All You Need Is Love†
Magical Mystery Tour	Fool On The Hill, Hello Goodbye, Penny Lane, All You Need Is Love†
The White Album	While My Guitar Gently Weeps, Blackbird, Piggies, Mother Nature's Son*
Let It Be	Two Of Us*, Across The Universe*, Let It Be, The Long And Winding Road, Get Back
Abbey Road	Come Together*, Something, Octopus's Garden, Here Comes The Sun*
Others (released as singles)	Love Me Do, From Me To You, She Loves You, I Want To Hold Your Hand, I Feel Fine, Day Tripper, We Can Work It Out, Paperback Writer, Lady Madonna, Hey Jude*, Don't Let Me Down*, The Ballad Of John And Yoko*

Table E.2: The catalogue of fifty melodies used in this dissertation.

*Music notation can be found for these melodies in Section E.4.

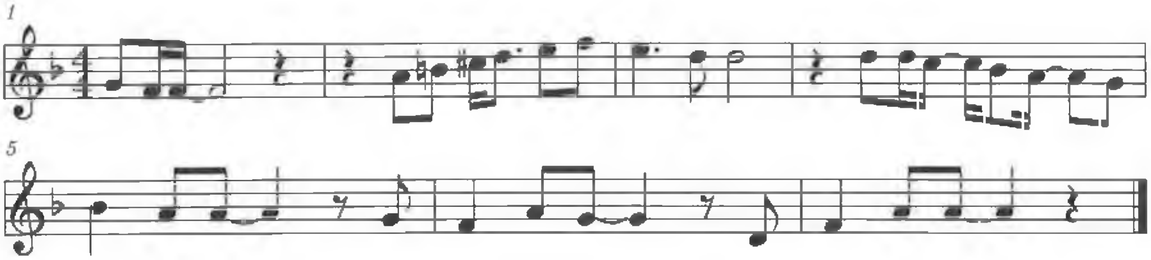
†These melodies were released on more than one album.

E.4 Illustrative samples in music notation

The melodies presented here are derived from Fujita et al. [53] and *The Beatles* original recordings. They are written an octave higher than they sound on these recordings. The tactus-spans were worked out by timing the recordings.

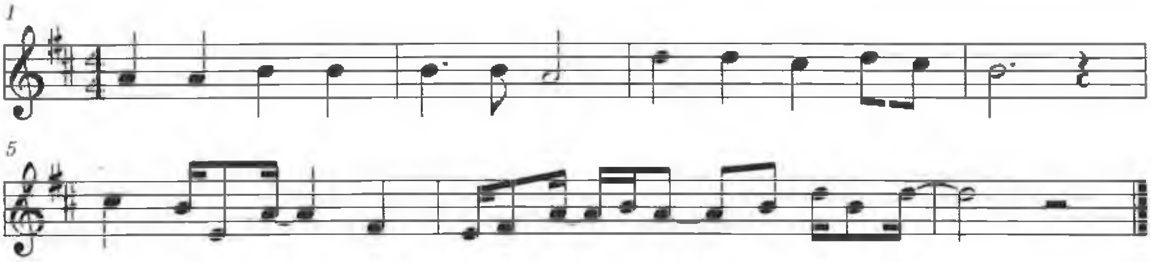
E.4.1 Yesterday

Written by John Lennon & Paul McCartney. The tactus-span is ♩ = 620ms.



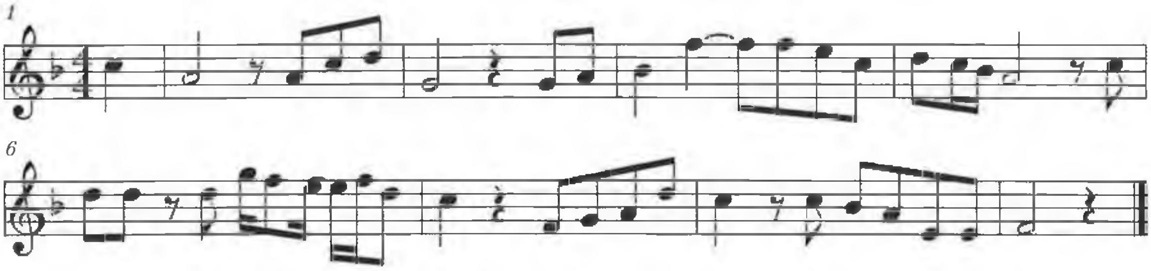
E.4.2 Mother Nature's Son

Written by John Lennon & Paul McCartney. The tactus-span is ♩ = 720ms.



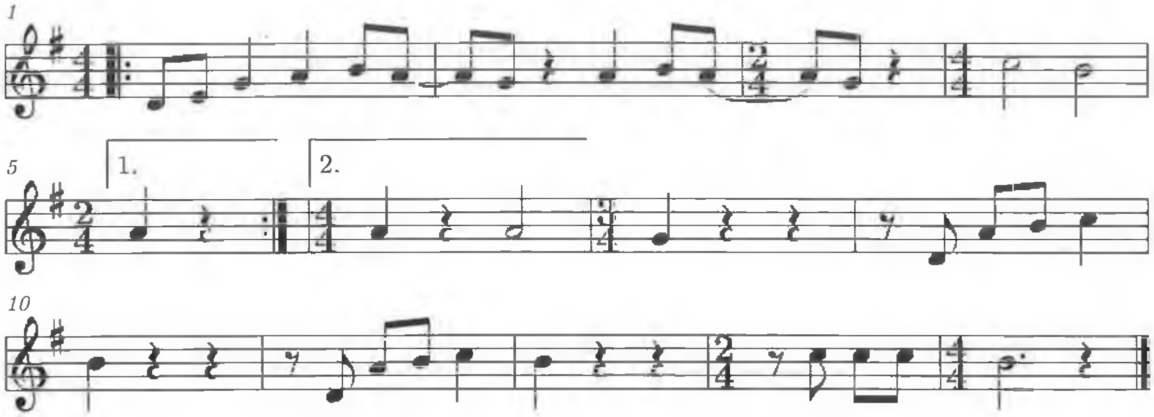
E.4.3 Hey Jude

Written by John Lennon & Paul McCartney. The tactus-span is ♩ = 800ms.



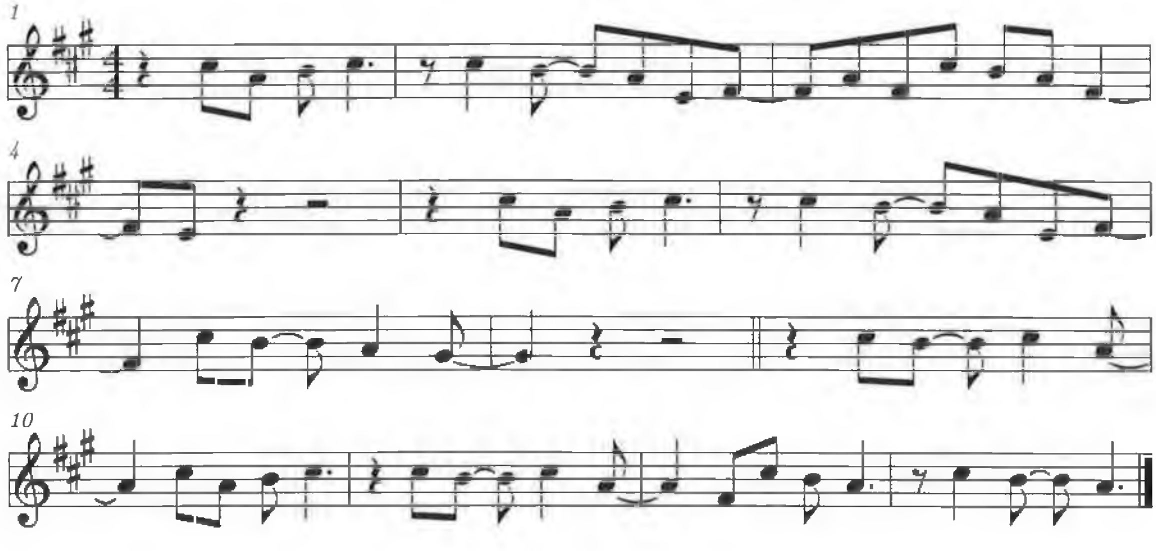
E.4.4 Two of Us

Written by John Lennon & Paul McCartney. The tactus-span is ♩ = $540ms$.



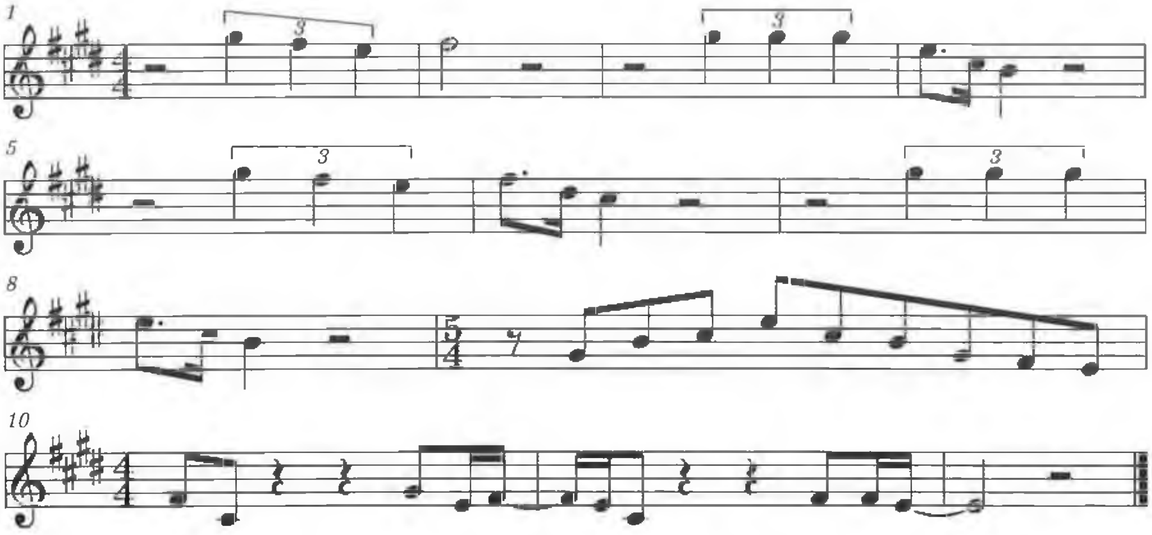
E.4.5 Here Comes the Sun

Written by George Harrison. The tactus-span is ♩ = $470ms$.



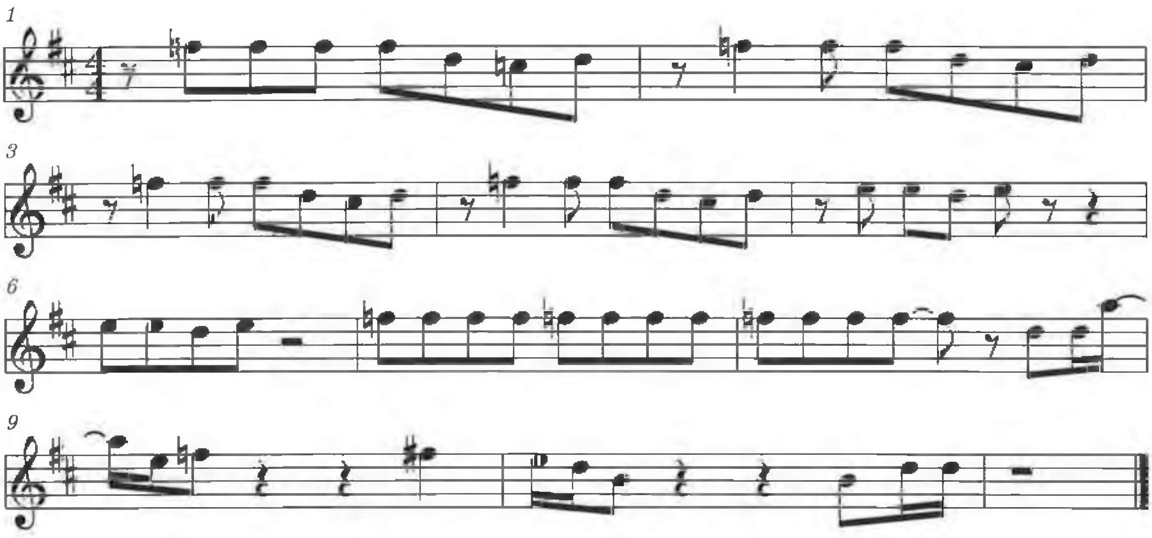
E.4.6 Don't Let Me Down

Written by John Lennon & Paul McCartney. The tactus-span is ♩ = 730*ms*.



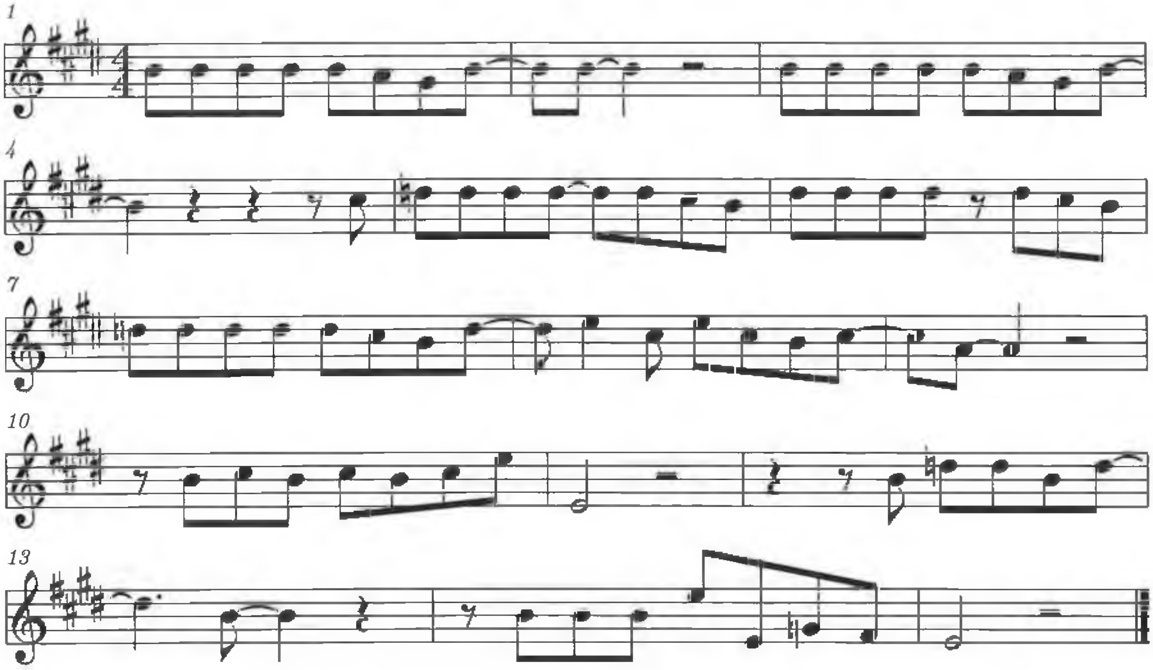
E.4.7 Come Together

Written by John Lennon & Paul McCartney. The tactus-span is ♩ = 730*ms*.



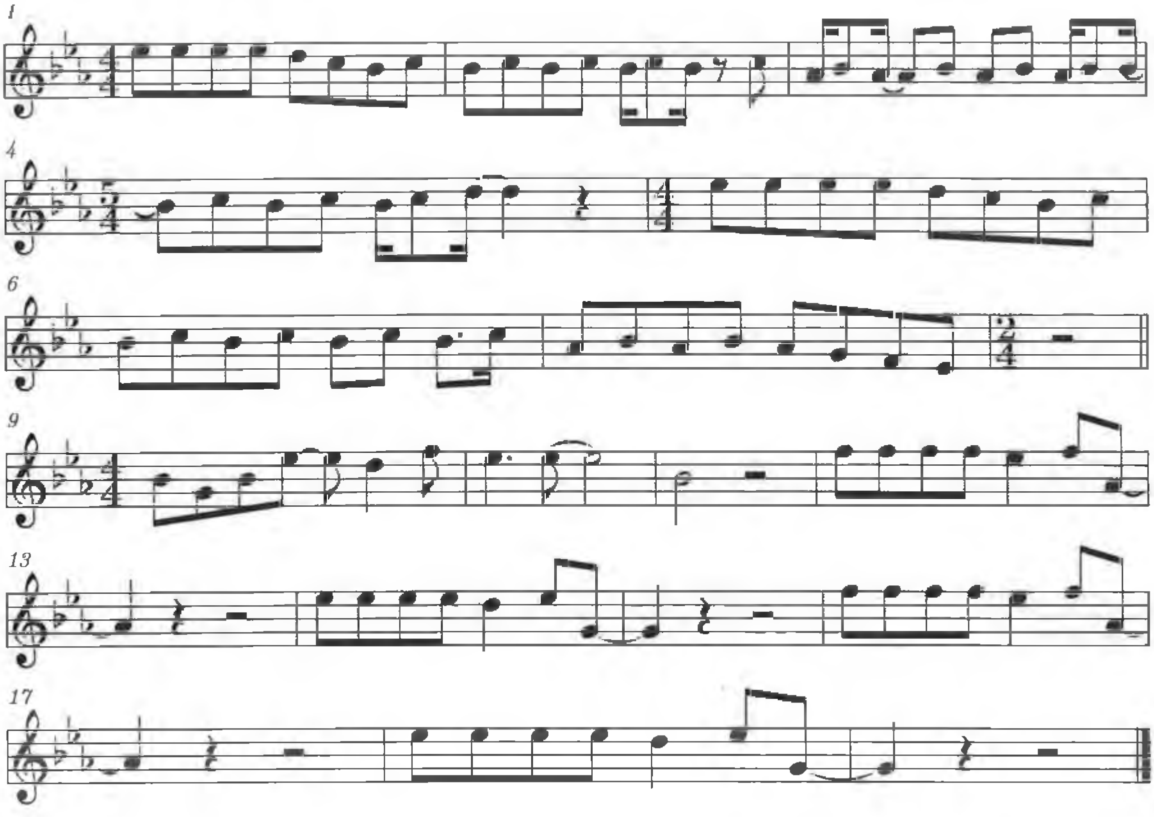
E.4.8 The Ballad of John and Yoko

Written by John Lennon & Paul McCartney. The tactus-span is ♩ = 440ms.



E.4.9 Across the Universe

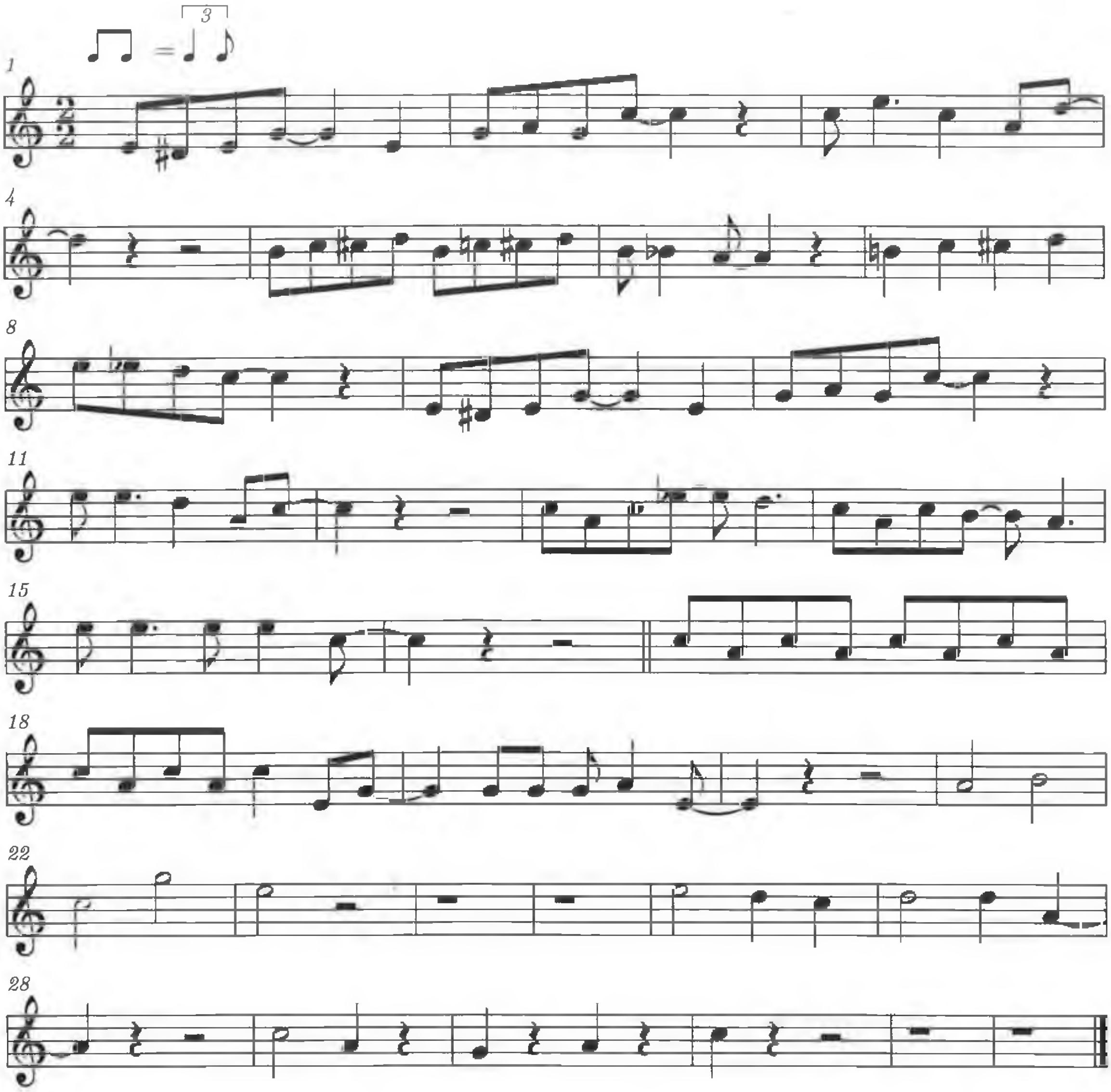
Written by John Lennon & Paul McCartney.[1] The tactus-span is ♩ = $725ms$.



[1] Originally, this melody was recorded in the key of *D major*. However, the version released on *Let It Be* was slowed down. This had the effect of bringing the song down a semitone to the key of *D♭ major*. Another version, released on *The World Wildlife Fund* charity album was actually sped up (this can be found on the *Past Masters: Volume 2* album). This had the effect of bringing the song up to the key of *E♭ major* [129, 158]. This latter version is presented here.

E.4.10 When I'm Sixty Four

Written by John Lennon & Paul McCartney.[2] The tactus-span is 𝅗𝅥 = 860ms.



[2] This melody is notated in *C major*, which is a semitone lower that the original recording in order to remove the proliferation of accidentals that would result if written in the original key of *Db major*.

Appendix F

Glossary of Parameters

A	Passive decay constant used during classification cell evolution.
B	Specifies the maximum value that a classification cell may reach.
$K_{althresh}$	A threshold above which associative learning should occur between cell assemblies at different SONNET modules within SONNET-MAP.
K_{chunk}	A boolean value that indicates whether or not the input cell associated with the smallest excitatory weight should be reset during the chunking-out process.
K_D	A small constant which specifies a floor value, allowing D_i to decrease gradually when c_i is very low.
K_{embed}	A predetermined threshold specifying the length z_{ni}^{tot} must reach before the relay-cell reset mechanism can be invoked.
K_{epoch}	Specifies the number of learning cycles for which an input sequence should be presented.
K_I	A link confidence threshold, above which a bottom-up link's normalized excitatory weight is considered to match its normalized input.
K_{inh}	Provides a ceiling on a SSG architecture's inhibitory weight learning formulation.
$K_{intervals}$	Used for interval encoding schemes (IOIs, pitch-intervals and contours for example). Specifies the range of intervals above which input cells are scaled by ω.
K_{latch}	The length of time a latch cell should remain active for, in response to an event, before being reset.
K_M	A threshold, above which a cell assembly is considered to match its current input.
K_{mode}	Specifies whether a SONNET module should operate in event-driven mode or time-driven mode.

Table F.1: Operational parameters used in the SONNET-MAP and *Re*TREE*ve* specifications (listed in alphabetical order).

$K_{overlap}$	The level of overlap tolerable before two cell assemblies can be considered as disjoint. Used for the CBR work.
$K_{recency}$	Specifies the number of additional beats at the end of an input sequence which should be processed.
$K_{\times}$	Constant used to determine the value of $I^{\times}_{ni}$. Previously known as K_2 (note that the K_1 and K_8 parameters are no longer required).
K_3	A threshold used to determine whether a secondary weight is large enough to become a member of the set T_{ni}.
K_4	The maximum length of the set T_{ni}. Used in event-driven SONNET configurations.
K_5	A z^{tot}_{ni} threshold, above which a cell assembly is considered to have a reasonable pattern representation and the learning rate modulator may be used.
K_7	A ceiling placed on the maximum inhibition that a cell assembly can receive.
K_9	Used in the event-driven inhibitory freezing mechanism. Specifies the percentage of a pattern that must be missing before that pattern is considered totally absent.
K_{10}	Specifies the minimum ratio (in % terms) between an inhibitory cell's activity and the activity of the most active inhibitory cell, before its classification cell may be reset.
K_{11}	Specifies the amount of time for which an input assembly should be active, before it can be considered for reset when the overload threshold K_{12} has been reached.
K_{12}	The level at which $\mathcal{F}_1$ is considered to have overloaded.
K_{13}	The activity level a classification cell must reach before it can be considered to have performed a classification.
K_{14}	The chunking-out delay. Specifies the period of time a cell assembly must remain above K_{13} before chunking-out can occur.
q	Specifies the minimum number of uncommitted cell assemblies that should respond to a given pattern.
T_{attn}	The period of time for which the network is allowed to change its state. Used only when SONNET operates in event-driven mode.
T_{span}	The tactus-span of the input sequence that is being processed. Used only when SONNET operates in time-driven mode.
T_{step}	Specifies the time step after which all network quantities are updated.
v_1	Fixed constant used during classification cell evolution.
v_2	Fixed constant used during classification cell evolution.
z_{min}	The minimum value that z^{tot}_{ni} may reach.

Table F.1: Operational parameters used in the SONNET-MAP and *Re*TREE*ve* specifications (listed in alphabetical order).

ϵ_1	Excitatory learning rate.
ϵ_2	Difference between the rates of change of excitatory and inhibitory learning.
ϵ_3	Used in the inhibitory weight learning equation for SSG architectures.
μ	The initial value to which an input cell activates in response to an event onset.
ρ	Vigilance parameter controlling the coarseness of classifications.
ω	The factor by which all active input cells continuously increase.

Table F.1: Operational parameters used in the SONNET-MAP and *Re*TREE*ve* specifications (listed in alphabetical order).

N_g	The number of sibling groups to be used at $\mathcal{F}_1$.
N_s	The number of siblings per group at $\mathcal{F}_1$.
N_l	The number of links between each $\gamma - \chi$ pair.
K_{zmin}	The minimum value which z^+_{mjni} may be initialized to.
K_{zmax}	The maximum value which z^+_{mjni} may be initialized to.
K_{winit}	The value which w^+_{mjni} is initialized to.
K_{uinit}	The value which u_{mjkl} is initialized to.

Table F.2: Architectural parameters used in the SONNET-MAP and *Re*TREE*ve* specifications.

Bibliography

[1] M. Alonso, G. Richard, and B. David. Tempo and beat estimation of musical signals. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[2] J. A. Anderson. *An Introduction to Neural Networks.* MIT Press, 1995.

[3] J. Aucouturier and F. Pachet. Music similarity measures: What's the use? In *Proceedings of the International Conference on Music Information Retrieval*, 2002.

[4] D. Bainbridge, S. J. Cunningham, and J. S. Downie. GREENSTONE as a music digital library toolkit. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[5] M. I. Bellgard and C. P. Tsang. Harmonizing music the boltzmann way. *Connection Science*, 6:281–297, 1994.

[6] J. J. Bharucha. Pitch, harmony, and neural nets: A psychological perspective. In P. M. Todd and D. G. Loy, editors, *Music and Connectionism.* MIT Press, 1991.

[7] J. J. Bharucha and P. M. Todd. Modeling the perception of tonal structure with neural nets. *Computer Music Journal*, 13(4):44–53, 1989.

[8] S. Blackburn and D. De Rouree. Musical part classification in content based systems. In *Proceedings of the International Workshop on Open Hypermedia Systems*, 2000.

[9] M. M. Botvinnik. *Computers in Chess: Solving Inexact Search Problems.* Springer-Verlag, 1984.

[10] W. E. Boyce and R. C. DiPrima. *Elementary Differential Equations.* Wiley, fifth edition, 1992.

[11] G. Bradski, G. A. Carpenter, and S. Grossberg. Working memories for storage and recall of arbitrary temporal sequences. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, 1992.

[12] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound.* MIT Press, 1990.

[13] E. M. Burns. Intervals, scales, and tuning. In D. Deutsch, editor, *The Psychology of Music.* Academic Press, second edition, 1999.

[14] G. A. Carpenter and S. Grossberg. ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930, 1987.

[15] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37: 54–115, 1987.

[16] G. A. Carpenter and S. Grossberg. ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, 3:129–152, 1990.

[17] G. A. Carpenter and S. Grossberg. ART 3 hierarchical search: Chemical transmitters in self-organizing pattern recognition architectures. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, 1990.

[18] G. A. Carpenter and S. Grossberg. Integrating symbolic and neural processing in a self-organizing architecture for pattern recognition and prediction. In V. Honavar and L. Uhr, editors, *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Academic Press, 1994.

[19] G. A. Carpenter and N. Markuzon. ARTMAP-IC and medical diagnosis: Instance counting and inconsistent cases. *Neural Networks*, 11:323–336, 1998.

[20] G. A. Carpenter and W. D. Ross. ART-EMAP: A neural network architecture for object recognition by evidence accumulation. *IEEE Transactions on Neural Networks*, 6:805–818, 1995.

[21] G. A. Carpenter, S. Grossberg, and J. H. Reynolds. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, 4:565–588, 1991.

[22] G. A. Carpenter, S. Grossberg, and K. Iizuka. Comparative performance measures of fuzzy ARTMAP, learned vector quantization, and back propagation for handwritten character recognition. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, 1992.

[23] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713, 1992.

[24] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen. Fuzzy ARTMAP: An adaptive resonance architecture for incremental learning of analog maps. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, 1992.

[25] G. A. Carpenter, S. Grossberg, and J. H. Reynolds. A neural network architecture for fast on-line supervised learning and pattern recognition. In H. Wechsler, editor, *Neural Networks for Perception: Human and Machine Perception*, volume 1. Academic Press, 1992.

[26] Ò. Celma, M. Ramírez, and P. Herrera. Foafing the music: A music recommendation system based on RSS feeds and user preferences. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[27] E. F. Clarke. Rhythm and timing in music. In D. Deutsch, editor, *The Psychology of Music*. Academic Press, second edition, 1999.

[28] M. A. Cohen and S. Grossberg. Masking fields: A massively parallel neural architecture for learning, recognizing, and predicting multiple groupings of patterned data. *Applied Optics*, 26:1866–1891, 1987.

[29] N. Collins. Using a pitch detector for onset detection. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[30] J. Copeland. *Artificial Intelligence: A Philosophical Introduction*. Blackwell, 1993.

[31] G. W. Cottrell, P. Munro, and D. Zipser. Image compression by back propagation: An example of extensional programming. In N. E. Sharkey, editor, *Advances in Cognitive Science*, volume 3. Norward, NJ: Ablex, 1988.

[32] R. B. Dannenberg. Toward automated holistic beat tracking, music analysis and understanding. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[33] M. Davies and M. Plumbley. Causal tempo tracking of audio. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[34] J. E. Dayhoff. *Neural Network Architectures: An Introduction*. Van Nostrand Reinhold, 1990.

[35] E. de Boer. *On the 'Residue' in Hearing*. PhD thesis, University of Amsterdam, 1956.

[36] D. Deutsch. The processing of pitch combinations. In D. Deutsch, editor, *The Psychology of Music*. Academic Press, second edition, 1999.

[37] D. Deutsch. Grouping mechanisms in music. In D. Deutsch, editor, *The Psychology of Music*. Academic Press, second edition, 1999.

[38] M. Dolson. Machine tongues XII: Neural networks. *Computer Music Journal*, 13(3):29–40, 1989.

[39] W. J. Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85:341–354, 1978.

[40] J. S. Downie. *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic n-grams as Text*. PhD thesis, University of Western Ontario, 1999.

[41] J. S. Downie. Toward the scientific evaluation of music information retrieval systems. In *Proceedings of the International Conference on Music Information Retrieval*, 2003.

[42] J. S. Downie. Music information retrieval. In B. Cronin, editor, *Annual Review of Information Science and Technology*, volume 37. Plexus Books, 2003.

[43] J. S. Downie, J. Futrelle, and D. Tcheng. The international music information retrieval systems evaluation laboratory: Governance, access and security. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[44] J. S. Downie, K. West, A. Ehmann, and E. Vincent. The 2005 music information retrieval evaluation exchange (MIREX 2005): Preliminary overview. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[45] D. Eck and N. Casagrande. Finding meter in music using an autocorrelation phase matrix and shannon entropy. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[46] G. M. Edelman. *Bright Air, Brilliant Fire: On the Matter of the Mind.* Penguin, 1992.

[47] C. H. Edwards and D. E. Penney. *Elementary Differential Equations with Applications.* Prentice Hall, third edition, 1994.

[48] J. Eggink and G. J. Brown. Extracting melody lines from complex audio. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[49] A. Eliëns. *Principles of Object-Oriented Software Development.* Addison-Wesley, 1994.

[50] J. L. Elman. Finding structure in time. 14:179–211, 1990.

[51] S. Franklin. *Artificial Minds.* MIT Press, 1995.

[52] K. Frieler. Beat and meter extraction using gaussified onsets. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[53] T. Fujita, Y. Hagino, H. Kubo, and G. Sato. *The Beatles Complete Scores.* Wise Publications, 1993.

[54] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[55] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: Musical information retrieval in an audio database. In *ACM Multimedia*, 1995.

[56] R. O. Gjerdingen. Using connectionist models to explore complex musical patterns. *Computer Music Journal*, 13(3):67–75, 1989.

[57] R. O. Gjerdingen. Apparent motion in music? In N. Griffith and P. M. Todd, editors, *Musical Networks: Parallel Distributed Perception and Performance.* MIT Press, 1999.

[58] J. L. Goldstein. An optimum processor for the central formation of pith of complex tones. *Journal of The Acoustical Society of America*, 54:1496–1516, 1973.

[59] N. Griffith. Development of tonal centres and abstract pitch as categorizations of pitch use. *Connection Science*, 6:155–175, 1994.

[60] N. Griffith and P. M. Todd, editors. *Musical Networks: Parallel Distributed Perception and Performance.* MIT Press, 1999.

[61] S. Grossberg. Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[62] S. Grossberg. Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, illusions. *Biological Cybernetics*, 23:187–202, 1976.

[63] S. Grossberg. Unitization, automaticity, temporal order, and word recognition. *Cognition and Brain Theory*, 7:263–283, 1984.

[64] S. Grossberg. Pitch-based streaming in auditory perception. In N. Griffith and P. M. Todd, editors, *Musical Networks: Parallel Distributed Perception and Performance.* MIT Press, 1999.

[65] S. Harford. Automatic segmentation, learning and retrieval of melodies using a self-organizing neural network. In *Proceedings of the International Conference on Music Information Retrieval*, 2003.

[66] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, second edition, 1999.

[67] R. N. A. Henson. Short-term memory for serial order: The start-end model. *Cognitive Psychology*, 36:73–137, 1998.

[68] H. H. Hoos, K. Renz, and M. Görg. GUIDO/MIR — an experimental musical information retrieval system based on GUIDO music notation. In *Proceedings of the International Conference on Music Information Retrieval*, 2001.

[69] A. J. M. Houtsma and J. L. Goldstein. The central origin of the pitch of complex tones: Evidence from musical interval recognition. *Journal of The Acoustical Society of America*, 51:520–529, 1972.

[70] M. I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Annual Conference of the Cognitive Science Society*, 1986.

[71] T. Kohonen. The "neural" phonetic typewriter. *Computer*, 21:11–22, 1988.

[72] T. Kohonen. *Self-Organizing Maps.* Springer, third edition, 2001.

[73] T. Kohonen, P. Laine, K. Tiits, and K. Torkkola. A nonheuristic automatic composing method. In P. M. Todd and D. G. Loy, editors, *Music and Connectionism.* MIT Press, 1991.

[74] B. Kolb and I. Q. Whishaw. *Fundamentals of Human Neuropsychology.* Freeman, fourth edition, 1996.

[75] A. Kornstädt. Themefinder: A web-based melodic search tool. In W. B. Hewlett and E. Selfridge-Field, editors, *Melodic Similarity: Concepts, Procedures, and Applications (Computing in Musicology 11).* MIT Press, 1998.

[76] C. L. Krumhansl. *Cognitive Foundations of Musical Pitch.* Oxford University Press, 1990.

[77] B. Laden and D. H. Keefe. The representation of pitch in a neural net model of chord classification. *Computer Music Journal*, 13(4):12–26, 1989.

[78] E. W. Large, C. Palmer, and J. B. Pollack. Reduced memory representations for music. In N. Griffith and P. M. Todd, editors, *Musical Networks: Parallel Distributed Perception and Performance.* MIT Press, 1999.

[79] O. Lartillot. Discovering musical pattern through perceptive heuristics. In *Proceedings of the International Conference on Music Information Retrieval*, 2003.

[80] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[81] M. Leman. The ontogenesis of tonal semantics: Results of a computer study. In P. M. Todd and D. G. Loy, editors, *Music and Connectionism.* MIT Press, 1991.

[82] K. Lemström and J. Tarhio. Transposition invariant pattern matching for multi-track strings. *Nordic Journal of Computing*, 10:185–205, 2003.

[83] K. Lemström, V. Mäkinen, A. Pienimäki, M. Turkia, and E. Ukkonen. The C-BRAHMS project. In *Proceedings of the International Conference on Music Information Retrieval*, 2003.

[84] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music.* MIT Press, 1983.

[85] D. J. Levitin. Memory for musical attributes. In P. R. Cook, editor, *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics.* MIT Press, 1999.

[86] D. Levy and M. Newborn. *How Computers Play Chess.* Computer Science Press, 1991.

[87] J. P. Lewis. Creation by refinement and the problem of algorithmic music composition. In P. M. Todd and D. G. Loy, editors, *Music and Connectionism.* MIT Press, 1991.

[88] P. Lewis. Using self-organizing neural networks to interpret melodic structures in the rhythm and pitch domains. Master's thesis, New Mexico Institute of Mining and Technology, 2002.

[89] B. Logan. Music recommendation from song sets. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[90] D. G. Loy. Connectionism and musiconomy. In P. M. Todd and D. G. Loy, editors, *Music and Connectionism.* MIT Press, 1991.

[91] M. Marolt. SONIC: Transcription of polyphonic piano music with neural networks. In *Proceedings of the Workshop on Current Research Directions in Computer Music*, 2001.

[92] J. A. Marshall and V. S. Gupta. Generalization and exclusive allocation of credit in unsupervised category learning. *Network: Computer Neural Systems*, 9:279–302, 1998.

[93] T. A. Marsland and J. Schaeffer, editors. *Computers, Chess, and Cognition.* Springer-Verlag, 1990.

[94] M. Mathews. The ear and how it works. In P. R. Cook, editor, *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics.* MIT Press, 1999.

[95] M. Mathews. The auditory brain. In P. R. Cook, editor, *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics.* MIT Press, 1999.

[96] J. L. McClelland, D. E. Rumelhart, and G. E. Hinton. The appeal of parallel distributed processing. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, 1986.

[97] M. McKinney and D. Moelants. Extracting the perceptual tempo from music. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[98] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proceedings of the ACM International Conference on Digital Libraries*, 1996.

[99] R. J. McNab, L. A. Smith, D. Bainbridge, and I. H. Witten. The new zealand digital library MELody inDEX. *D-Lib Magazine*, 3, 1997.

[100] C. Meek and W. Birmingham. Johnny can't sing: A comprehensive error model for sung music queries. In *Proceedings of the International Conference on Music Information Retrieval*, 2002.

[101] M. Melucci and N. Orio. A comparison of manual and automatic melody segmentation. In *Proceedings of the International Conference on Music Information Retrieval*, 2002.

[102] S. Meyers. *More Effective C++: 35 New Ways to Improve Your Programs and Designs.* Addison-Wesley, 1996.

[103] S. Meyers. *Effective C++: 50 Specific Ways to Improve Your Programs and Designs.* Addison-Wesley, second edition, 1998.

[104] M. Minsky. *The Society of Mind.* Simon & Schuster, 1985.

[105] M. C. Mozer. Connectionist music composition based on melodic, stylistic, and psychophysical constraints. In P. M. Todd and D. G. Loy, editors, *Music and Connectionism.* MIT Press, 1991.

[106] M. C. Mozer. Neural net architectures for temporal sequence processing. In A. S. Weigend and N. A. Gershenfeld, editors, *Times Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1993.

[107] M. C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6:247–280, 1994.

[108] G. Neve and N. Orio. Indexing and retrieval of music documents through pattern analysis and data fusion techniques. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[109] G. Neve and N. Orio. Experiments on segmentation techniques for music documents indexing. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[110] A. Nigrin. The real-time classification of temporal sequences with an adaptive resonance circuit. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, 1990.

[111] A. Nigrin. SONNET: A self-organizing neural network that classifies multiple patterns simultaneously. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, 1990.

[112] A. Nigrin. *The Stable Learning of Temporal Patterns with and Adaptive Resonance Circuit.* PhD thesis, Duke University, 1990.

[113] A. Nigrin. A new architecture for achieving translational invariant recognition of objects. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, 1992.

[114] A. Nigrin. *Neural Networks for Pattern Recognition.* MIT Press, 1993.

[115] J. Nolte. *The Human Brain: An Introduction to Its Functional Anatomy.* Mosby, fourth edition, 1999.

[116] J. L. Noyes. *Artificial Intelligence with Common Lisp: Fundamentals of Symbolic and Numeric Processing.* Heath, 1992.

[117] N. Orio and M. S. Sette. A HMM-based pitch tracker for audio queries. In *Proceedings of the International Conference on Music Information Retrieval*, 2003.

[118] M. P. A. Page. *Modelling Aspects of Music Perception using Self-Organizing Neural Networks.* PhD thesis, University of Wales, 1993.

[119] M. P. A. Page. Modelling the perception of musical sequences with self-organizing neural networks. *Connection Science*, 6:223–246, 1994.

[120] B. Pardo. Tempo tracking with a single oscillator. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[121] J. Paulus and A. Klapuri. Measuring the similarity of rhythmic patterns. In *Proceedings of the International Conference on Music Information Retrieval*, 2002.

[122] S. Pauws. CubyHum: A fully operational query by humming system. In *Proceedings of the International Conference on Music Information Retrieval*, 2002.

[123] G. Peeters. Rhythm classification using spectral rhythm patterns. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[124] A. Pienimäka. Indexing music databases using automatic extraction of frequent phrases. In *Proceedings of the International Conference on Music Information Retrieval*, 2002.

[125] J. R. Pierce. The nature of musical sound. In D. Deutsch, editor, *The Psychology of Music*. Academic Press, second edition, 1999.

[126] J. R. Pierce. Sound waves and sine waves. In P. R. Cook, editor, *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*. MIT Press, 1999.

[127] J. R. Pierce. Introduction to pitch perception. In P. R. Cook, editor, *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*. MIT Press, 1999.

[128] A. Pikrakis, I. Antonopoulos, and S. Theodoridis. Music meter and tempo tracking from raw polyphonic audio. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[129] A. W. Pollack. Notes on "Across The Universe": Notes on ... series no. 169. The 'Official' rec.music.beatles Home Page, 1999. URL `http://www.recmusicbeatles.com`.

[130] L. Prechelt and R. Typke. An interface for melody input. *ACM Transactions on Computer-Human Interaction*, 8:133–149, 2001.

[131] R. Rasch and R. Plomp. The perception of musical tones. In D. Deutsch, editor, *The Psychology of Music*. Academic Press, second edition, 1999.

[132] E. Rich and K. Knight. *Artificial Intelligence*. Mc Graw-Hill, second edition, 1991.

[133] S. C. Roberts. *Interpreting Rhythmic Structures using Artificial Neural Networks*. PhD thesis, University of Wales, 1996.

[134] S. C. Roberts and M. Greenhough. The detection of rhythmic repetition using a self-organising neural network. In *Proceedings of the International Computer Music Conference*, 1994.

[135] S. C. Roberts and M. Greenhough. Rhythmic pattern processing using a self-organising neural network. In *Proceedings of the International Computer Music Conference*, 1995.

[136] M. A. Rubin. Application of Fuzzy ARTMAP and ART-EMAP to automatic target recognition using radar range profiles. *Neural Networks*, 8:1109–1116, 1995.

[137] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, 1986.

[138] D. E. Rumelhart and J. L. McClelland, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. MIT Press, 1986.

[139] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, 1986.

[140] H. Sano and B. K. Jenkins. A neural network model for pitch perception. *Computer Music Journal*, 13(3):41–48, 1989.

[141] D. L. Scarborough, B. O. Miller, and J. A. Jones. Connectionist models for tonal analysis. *Computer Music Journal*, 13(3):49–55, 1989.

[142] J. F. Schouten. The residue and the mechanism of hearing. In *Proceedings Koninklijke Nederlandse Akademie van Wetenschappen*, volume 43, 1940.

[143] M. Schuppert, T. F. Münte, B. M. Wieringa, and E. Altenmüller. Receptive amusia: Evidence for cross-hemispheric neural networks underlying music processing strategies. *Brain*, 123: 546–559, 2000.

[144] R. Sedgewick. *Algorithms in C++*. Addison-Wesley, 1992.

[145] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

[146] A. Shaw and R. A. Mitchell. Phoneme recognition with a time-delay neural network. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, 1990.

[147] R. Shepard. Geometric approximations to the structure of musical pitch. *Psychological Review*, 89:305–333, 1982.

[148] R. Shepard. Stream segregation and ambiguity in music. In P. R. Cook, editor, *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*. MIT Press, 1999.

[149] R. Shepard. Tonal structure and scales. In P. R. Cook, editor, *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*. MIT Press, 1999.

[150] E. H. Shortliffe. *Computer-Based Medical Consultations: MYCIN*. Elsevier, 1976.

[151] B. Snyder. *Music and Memory: An Introduction*. MIT Press, 2000.

[152] T. Sødring. *Content-Based Retrieval of Digital Music.* PhD thesis, Dublin City University, 2002.

[153] M. W. Spratling. Pre-synaptic lateral inhibition provides a better architecture for self-organizing neural networks. *Network: Computer Neural Systems*, 10:285–301, 1999.

[154] L. R. Squire and E. R. Kandel. *Memory: From Mind to Molecules.* Scientific American Library, 2000.

[155] I. Taylor and M. Greenhough. Modelling pitch perception with adaptive resonance theory artificial neural networks. *Connection Science*, 6:135–154, 1994.

[156] E. Terhardt. Zur tonhoehenwahrnehmung von klaengen II: ein funktionsschema. *Acustica*, 26:187–199, 1972.

[157] E. Terhardt, G. Stoll, and M. Seewann. Algorithm for extraction of pitch and pitch saliences from complex tonal signals. *Journal of The Acoustical Society of America*, 71:679–688, 1982.

[158] G. Tillekens. A Beatles' odyssey, Alan W. Pollack's musicological journey through the Beatles' songs. *Soundscapes — Journal on Media Culture*, 1, 1999.

[159] P. M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.

[160] P. M. Todd and D. G. Loy, editors. *Music and Connectionism.* MIT Press, 1991.

[161] P. Toiviainen and T. Eerola. Classification of musical metre with autocorrelation and discriminant functions. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[162] R. Typke, F. Wiering, and R. C. Veltkamp. A survey of music information retrieval systems. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[163] E. Ukkonen, K. Lemström, and V. Mäkinen. Sweepline the music! In R. Klein, H. Six, and L. Wegner, editors, *Computer Science in Perspective.* Springer-Verlag, 2003.

[164] H. L. F. von Helmholtz. *On the Sensations of Tone as a Physiological Basis for the Theory of Music.* Dover, 1954. Originally published in 1863, English translation by A. J. Ellis.

[165] B. De Vries and J. C. Principe. The gamma model — a new neural model for temporal processing. *Neural Networks*, 5:565–576, 1992.

[166] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:328–339, 1989.

[167] E. A. Wan. Time series prediction by using a connectionist network with internal delay lines. In A. S. Weigend and N. A. Gershenfeld, editors, *Times Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1993.

[168] D. Wang and M. A. Arbib. Complex temporal sequence learning based on short-term memory. *Proceedings of the IEEE*, 78:1536–1543, 1990.

[169] D. Wang and M. A. Arbib. Timing and chunking in processing temporal order. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:993–1009, 1993.

[170] P. D. Wasserman. *Neural Computing: Theory and Practice.* Van Nostrand Reinhold, 1989.

[171] N. M. Weinberger. Music and the auditory system. In D. Deutsch, editor, *The Psychology of Music.* Academic Press, second edition, 1999.

[172] T. Weyde. The influence of pitch on melodic segmentation. In *Proceedings of the International Conference on Music Information Retrieval*, 2004.

[173] F. L. Wightman. The pattern-transformation model of pitch. *Journal of The Acoustical Society of America*, 54:407–416, 1973.

[174] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[175] R. J. Williams and D. Zipser. Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1:87–111, 1989.

[176] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin and D. E. Rumelhart, editors, *Backpropagation: Theory, Architectures and Applications.* Erlbaum, 1995.