THE EAR'S MIND

A COMPUTER MODEL OF THE FUNDAMENTAL MECHANISMS OF THE PERCEPTION OF SOUND

Ramon Dor

24 August 2005

Delft University of Technology
Faculty of Electrical Engineering,
Mathematics and Computer Sciences

THE EAR'S MIND

A COMPUTER MODEL OF THE FUNDAMENTAL MECHANISMS OF THE PERCEPTION OF SOUND

Graduation committee:

Dr. drs. L. J. M. Rothkrantz, thesis supervisor

Dr. E. A. Hendriks

Dr. J. Jensen

Ramon Dor, student nr. 9194233

Master thesis, 24 August 2005

THE EAR'S MIND

Delft University of Technology

Faculty of Electrical Engineering,

Mathematics and Computer Science

Mediamatics group

TABLE OF CONTENTS

Abstract

A novel computer model is proposed for the primitive perception of sound. The model, titled *The Ear's Mind* is principally based on the *Copycat* computer model (Mitchell 1993, Hofstadter 1995). *The Ear's Mind* is designed to model the unconscious, automatic auditory grouping pressures in humans. Such pressures, it seems, steer the perception of sound by cooperative and competitive interactions, resulting in the grouping of sound elements into context-sensible entities (see Bregman 1990, Rosenthal 1998, Ehret 1997). The auditory aspects of the proposed model are based on psychoacoustic experiments taken from the field of *Auditory Scene Analysis* (Bregman 1990). Like a community of social insects, where high-level phenomena emerge from local activities at lower-levels in a process of self-organisation, an interpretation of the auditory scene conforming human auditory perception is meant to emerge from the local activities of *The Ear's Mind*'s agents. Although the current implementation is tailored for the auditory domain, the underlying theory of *The Ear's Mind* is meant as a general architecture for emergent sensory perception. Preliminary results support the theory and show both flexible context-sensitivity and the ability to match simple grouping behaviour. Further model expansion is planned.

Acknowledgements

The time and effort I put into this work could not have been done without affecting others. And in most cases, these effects meant that others had to put up with my lack of time. I'd like to thank my friends, musicians, professors of music and others from the music world that were patient and flexible enough to cope with my erratic juggling lifestyle between the arts and science. From a totally different perspective, I'd like to thank my friends, scientists, professors of science and others from the scientific world that were patient and flexible enough to cope with my erratic juggling lifestyle between science and the arts.

To D. Hofstadter, I am grateful for allowing me to enter the fascinating world of cognitive science through your writings, and showing me how things look through your Mind's I. To Emil Hendriks, thank you for your patience and encouragement throughout the years, 'yes, I'm still alive…' And finally, to Leon Rothkrantz, my thesis supervisor, who has always found the time for our inspiring meetings, which seemed to cover anything from Fourier to Bach and from social insects to piano technique.

PART ONE: FOUNDATIONS

1 Introduction

This work is about the mind. But not the kind of mind we normally think of. Never mind conscious thought, learned behaviour and the representation of knowledge. It's about a mind at a much lower level. Mind you, without it, one would probably only mind one's own business. What else can you do without perceiving the world outside you? For every sense, there are cognitive processes at the back of one's mind of which one is never aware. They 'micro'-perceive the world and present us with the objects we believe we see and hear. It's as if they have a mind of their own, at least at some level, where different strategies, opinions and pressures cooperate or obstruct each other, and come up with untangling interpretations for the constant stream of knotty sensory input. They make up our minds. And if they do have their own minds, then just as we talk about the Mind's eye, just as we talk about the Mind's I, we can talk about the Eye's Mind. This work is about these sorts of minds. What are they? How do they work? Can we simulate them? In particular, it's about a computer model for the mechanisms that bring about the perception of sound. And this is done by The Ear's Mind.

Goals

In our day and age, it seems that computers have conquered and automated every aspect of human life. In this context, however, it is remarkable to realise that computers have not yet reached even the most modest capabilities of primitive sensory perception. Moreover, I believe that this unsatisfactory state is related to basing computer models of perception on the non-biologically motivated centralised computation paradigm. It is therefore a central goal of this work to study whether computational emergent mechanisms may help bringing us closer to better functionality. Consequently, and for theoretically motivating this work, it was my intention to survey existing evidence for supporting the use of emergence for sensory perception. This entails investigating how well evidence from the fields of cognitive psychology and psychoacoustics in general, auditory scene analysis, and neuroanatomy fit such interpretations. I therefore aim at designing and testing a system, which attempts to mimic the perceptual grouping processes of the human ear as witnessed by data from psychoacoustics. This system has to comply with functional constraints from relevant fields of research ranging from biological and evolutionary models to neurobiology and self-organising complex systems. It is my belief, and therefore the main pillar of this work, that the best way to mimic nature's complex adaptive processes is to follow her example.

In this work, I propose a theoretical model for primitive sensory perception. Based on the proposed model, a computer programme, tailored for auditory perception, will be implemented. It is the intention that the preliminary implementation of the proposed model will show basic capabilities of grouping auditory features extracted from input sound fragments matching those of human subjects in psychoacoustic experiments. Consequently, the model will be subjected to input from standard psychoacoustic experiments and the grouping results will be compared with

those of human subjects. Although the objective is to come as close as possible to human grouping tendencies, the proposed model is viewed as a testing ground for the general underlying theory of implementing primitive perception by self-organisation. Great effort was spent on ensuring that the implementation phase of the theory will not stand in the way of future model expansion. As a result, the model as a whole can be used as a self-contained laboratory for experimenting with different types of grouping pressures. Due to the large extent of the project, this work will describe the first phase – the proof of concept phase, which aims to implement a minimal set of mechanisms capable of falsifying the proposed theory. That is, proving the theory wrong if results fail expectations. If the model passes preliminary tests, however, and the theoretical approach survives, work would continue by adding functionality to the already implemented foundation.

Motivation

As a professional musician, I was always intrigued by the complexity of the perception of sound, trying to distil the processes responsible for hearing what we hear from what is actually being played. Performers and Composers of Music alike rely on the differences between the real and the perceived to great extents. These phenomena range from the simple illusions of legato playing, the use of contrasting articulations for bringing out melodic lines, to blending orchestral parts into coherent colours or minutely altering the timing of melodies to separate them from the accompaniment. We add tiny accents to make a passage sound even, and when we let the computer play it 'perfectly' it may even sound un-rhythmic to our ears. I was even more intrigued when reading Bregman's Auditory Scene Analysis (Bregman 1990) and learning of the many psychoacoustic experiments showing the existence of what seemed to be simple grouping pressures. Such pressures unceasingly work behind the scenes, steering our perception of sound according to the surrounding context.

From a very different angle, I have always been fascinated with the cognitive sciences, and theories of mind. Perhaps the biggest experience I had in this field was when I was 'hit' by the fabulously rich work by D. Hofstadter, titled *Gödel, Escher, Bach: An Eternal Golden Braid (Hofstadter 1979).* It was a fascinating experience to be led through Hofstadter's narrative of theories of mind, and ant communities. From music to math and back all in the best story-telling manner, I was captured. From that moment on, I was constantly trying to imagine how such 'cognitive communities' could be emulated by computers. Reading further on complex behaviours emerging from local interactions of anything from chemicals and biological complex systems to mass human behaviour, I became convinced that the principle of letting such systems come up with context sensitive solutions to complex problems is employed everywhere you look at in the natural world. From an evolutionary perspective, it seems to be an extremely efficient way to achieve flexibility of behaviour without designing it from scratch. And if evolution makes use of these processes, then there is no reason why it should restrict them to outer-cranium activities.
Hofstadter's *Fluid concepts and creative analogies (Hofstadter 1995),* containing a description of his computer models of analogy making, it struck me that the use of analogies in Hofstadter's Copycat was analogous to my understanding of analogy as a means of grouping sensory entities together. Simply put, what context sensitive primitive perception boils down to, is grouping 'the same' entities together. Only that, 'the same' is not 'really' the same. Rather, it is an analogy that holds in a certain context. An analogy that emerges from a 'soup' of local pressures, each pulling its own way. The Copycat computer model seemed to contain many of the mechanisms needed for the implementation of a preliminary 'proof of concept' model for auditory perception. Consequently, I based *The Ear's Mind* foundations on the Copycat architecture altering, and adding mechanisms to fit the theory and the auditory domain.

I start with the necessary background for understanding the auditory grouping processes which the model attempts to emulate. In ***Chapter 2, Auditory Scene Analysis,*** I briefly cover the available data regarding the perception of sound in general, and the primitive grouping mechanisms that are believed to be in charge of organising the auditory scene into separate auditory objects. I argue that these processes lead to an *interpretation,* not *detection* of the world around us, a necessary consequence of the limited amount of auditory evidence, and the way in which sound cues interfere with each other, corrupting the available information originally contained in each auditory object in isolation. The chapter introduces and discusses the auditory scene, the nature of the primitive elements of sound, auditory sensory periphery and neuroanatomy, and psychoacoustics.

Chapter 3, Computational Auditory Scene Analysis, covers significant attempts of modelling different aspects of Auditory Scene Analysis by computational means. This chapter is included for three reasons. First, by telling the story of some attempts at cracking the auditory perception problem, it introduces the reader to the difficulties of building working models of sound perception. Second, by analysing the pitfalls, one can pursue alternative pathways. Third, and by no means less important, one has to guard at all times not to hit the snag of modelling only a single aspect of sound perception. Assuming that later expansions and interaction with processing at other levels will solve the very problems keeping the current model from functioning in the real world is a grave mistake.

The Ear's Mind is an emergent system, which works as a non-supervised collection of independent local primitive agents. These agents build or destroy bridges in the data-landscape they work on. It is a stochastic model principally based on the Copycat model. An introduction to Copycat and a discussion of its significance to perception mechanisms and *The Ear's Mind* is given in ***Chapter 4, Analogy Making As Perception***.

Chapter 5, Requirements for a computational model of perception discusses the constraints and design strategies that I believe to be of importance for the implementation of computational models of perception. Evidence from past chapters is used to derive such constraints from the fields of psychoacoustics, anatomy, computational auditory scene analysis, biology and psychology. This is by no means an attempt to do the impossible and cover all available knowledge from all the different fields. Rather, this chapter is a checklist meant for conforming with constraints from different fields pertaining to auditory perception. Following this, I propose in ***Chapter 6, Copycat and (auditory) perception*** that the Copycat model fits many of the constraints listed in chapter 5, and list some of its additional properties that make it a good candidate on which *The Ear's Mind* could be based.

Chapter 7, The Ear's Mind: implementation introduces the proposed model. Starting with a discussion of the model as a proof of concept for other domains of sensory perception and a comparison between the auditory domain and Copycat's original domain, I follow with the implementation details of the model itself. This includes porting the Copycat model into a modern dialect, choosing implementation languages, and implementing the new modules including a preprocessor, a graphics server and domain specific mechanisms.

Finally, in ***Chapter 8, The Ear's Mind: performance*** I present some of the experiments and tests I have done with the implemented model, and discuss the results. Plans for future work and conclusions are included at the end of the chapter.

2 Auditory Scene Analysis

The Perceptual Organization of Sound

We use our senses as gateways to the world around us. Through such peepholes as our ears and eyes, we build an inner model of the world. The way I see it, we have no other choice but to believe that this model ***is*** *actually the real world. Such mirroring from outer- to inner-reality is so innate that it is nearly impossible for us to catch ourselves cheating. Seeing a familiar face, we awaken that face up in our inner-world; when talking to the person owning that face we send messages through to the outside - to our conception of the outside. This behaviour feels so natural, that it reveals itself only when it breaks down. It is only when attempting to emulate even the simplest capabilities of perception by computational means - when witnessing the difficulties of those with agnostic failures such as aphasia and amusia - only then we realize the huge gap between our* ***perception*** *of the real-world (our inner-world), and the* ***real*** *world outside ourselves [1]. Another way of appreciating the incredible transformations taking place between the outer- and inner-world, real- and model-world is by experiencing any of the vast collection of paradoxes with which we can literally fool our own senses.*

It is by ***interpretation*** *of sensory input, that we are capable of building an inner model of our outer-cranial-scene. But interpretation does not mean getting it right all the time; it doesn't even mean knowing for certain that a perceived event has actually taken place out there, even though we may not be conscious of that uncertainty. Real-world events, after all, stay outside. It is only the evidence we collect, which we put together like the pieces of a jigsaw-puzzle that forms a sensible, acceptable account of what we eventually believe 'really' happened. Though visual input is just as difficult (or even tougher) to interpret as auditory- or tactile-input, I chose to concentrate here on the interpretation of the auditory world scene. The field of research dealing with the interpretation of the auditory evidence entering our heads from the outside-scene is called Auditory Scene Analysis.*

2.1 The auditory scene

The work of researchers in the fields of Auditory Scene Analysis (ASA) and Computational Auditory Scene Analysis (CASA) would certainly be much easier if the structures of acoustic events (such as an animal call, or the blowing of the wind) were simple, and allowed the *direct* activation of the corresponding concepts in our mind's I. As it is, though, the evidence we receive is much more complex than the auditory event it represents. Here is an old (several hundred million years) incomplete 'wish list' of an average creature with the expectations of evolving the powers of auditory scene analysis sometime in the next hundred million years:

- I wish that every sound event would be simple, that is, non-compound. As it is, the vast majority of sounds are composite events, each containing several harmonic- and noise-components and are almost always composed of temporal sequences.

[1] Implicitly, for the purposes of this work, I accept the events which we perceive as real. That is, that if we correctly perceive sound entering our ears, that that sound has actually taken place in the physical world, and has propagated through a suitable medium to reach our ears.

- It would help if evidence from distinct auditory events would stay distinct when it reached our ears. However, when entering our ears pieces of evidence from distinct sound-events blend and resemble none of the original events producing it. Now we have different cues from different events either side by side or occluding each other.

- Things could be much simpler if each event would reach the ears only once. In the real world, however, sound waves are unceasingly reflected by any object, sending delayed copies to our ears.

- If evidence only from 'real' events would be received, the task of grouping cues together would be much easier. Unfortunately, enough cues from events such as random noise and pseudo-cues arising from imperfections of the auditory system itself get mixed with all of the above.

Now we have to take all this evidence and compose a world-view, an image of what might have taken place just a fraction of a second ago in the real world. Fast enough to be able to react on it. To this day, no computer algorithm has even come close to negotiating such difficulties. It is the aim of *Computational Auditory Scene Analysis (CASA)* researchers to build computer models which tackle such problems. The two main routes of investigation are:

- Treating sound events as data and perception mechanisms as a deterministic bottom-up information processing system. Tackling the problem using mathematical (blind algorithms, statistical signal processing, etc.) means of solving the problem.

- Consulting nature: how do humans and other animals do it? We can learn how to imitate solutions biological systems have found.

We have spent millions of years perfecting the skills of auditory perception. It is my belief that following nature (though it may sometimes lead to more questions than answers) will prove more fruitful at the end. Here are two whys. More details can be found throughout this work.

- Since perception is an *interpretation*, not *detection* of reality, it follows that one should follow the interpretive mechanisms of perception. The idea is to build a model of what we perceive of reality, not of reality itself. All applications of speech processing, and production, for instance, rely on the ability to imitate the way *we* hear speech, not only the physics of sound. For example, the mp3 compression scheme relies on the psychoacoustic model of masking. Ignoring *us* by following pure mathematical means may very well lead to the construction of effective information-processing tools for all sorts of handy tasks, but it will not get us closer to perceiving the way *we* do it. If you want a computer to 'hear' what we hear, or even to segment sound the way we do it, you have to keep in mind that what we hear is a subjective interpretation we make of the outside scene, nothing even close to the objective events which actually took place.

- Auditory perception is critical for evolutionary survival. The evolution of perception went hand in hand with the evolution of human and animal sound-production as a means of communication. Sound events take on an ecological context. It follows that our perception mechanisms are specially tailored for us; we need to know the way we do it to imitate the way we perceive.

In the following section, I cover some of the basic mechanisms animals, including humans, use for the analysis of auditory scenes.

2.2 Auditory cues

When presented with so many pieces of evidence, all jumbled up together, how can we know which pieces are relevant, which are more important? How do we even know what we're looking for? Many computer models are designed for processing input sounds with the use of *a priori* information such as the number of human voices existing in a recorded audio fragment. This makes it a much easier task. The programme knows what to look for in advance. However, such models break down as soon as no prior information is available. What a priori knowledge can we use, when the nature of the input is unknown? The following exemplifies some of the cues humans use for segmenting sounds.

- Compound auditory events impose a common behaviour (or fate, see section 5.3, Gestalt) on their constituent cues. When a compound event changes in frequency, (as speech-vowels or bird songs do) then all its components will follow roughly the same transformation. Cues having nearly the same on-set times are very likely to have been originated from the same object, rather than from unrelated events.

- Cues differing in their respective steady states will tend to segregate from each other. Cues arriving from different locations (e.g. having different inter-aural delays) suggest that they originate from two distinct events at distinct locations.

- Gradual changes in the sound data (such as small temporal intensity gradients) would indicate a change in the sound produced by an existing object. Abrupt changes might be interpreted as the beginning of a new event.

- Successive sound having similar spectral shapes will tend to be grouped together into a so-called auditory stream. Such a stream is interpreted as successive sounds originating from the same entity in the real world.

- Differences in global properties (such as the positions of spectral peaks and valleys, intensity etc.), could lead to the segregation of cues.

- When new on-set cues appear in an occupied spectral range of an already existing object, there is reason to believe that the old one still continues 'behind' the occlusion of a new object.

- Since many sounds in the physical world in general and animal calls in particular produce harmonic spectral components, it should be a good idea to group together those simultaneous components which happen to be harmonically related to one another.

Results of extensive research in the field of auditory scene analysis (Bregman 1990) give compelling evidence to support a theory of perception based on the use of a whole array of heuristics such as the above list. As one might expect, there are many questions left unanswered pertaining to the exact implementation of such a heuristic scheme; a few questions, however, are so critical, that without finding proper answers for them, no artificial perception machine could do anything interesting. These questions have to do with the architecture in which all the different heuristics are put to work. They also have to do with defining the primitive units of cues, and how such units can blob together to form larger, more abstract cues. Other open questions are: How do we know if successive sounds originate from a single object, when sounds from other objects are interleaved in the sequence? To what extent processes of a much higher level (due to learning, or attention) take charge of perception; can we perceive speech only because we have learned to recognise it? Can an infant group melodies the first it hears music?

What kind of melodies can it group? What kind of analogies can it recognise? Do we expect a computer programme which 'masters' segmentation heuristics to be able to group together simple melodies? Separate speech from interfering background noises?

Take for example a singer accompanied by a guitar. On-set cues (detectors of strong temporal intensity derivatives) support harmonic cues (detectors of harmonic relations between spectral components) and common frequency modulation cues (shared between all harmonics due to global voice vibrato) and consequently form a new unit - a complex tone - segregating from all other cues in the spectro-temporal vicinity. The new tone, now a unit in its own right attaches to some other complex tones preceding it. If the 'same' tone is sung again, we recognise it as such, even though the cues for the two tones might be very different. Possibly, a new unit will be formed coalescing successive tones to form a melody; simultaneous notes will coalesce to form harmony. Notes might take part in more than several such units (e.g. melody, harmony, rhythm). Analogies between larger units will parse the streams into grammar-like structures. The complexities are unbelievable. How about separating simultaneous guitar and voice notes, which share the same pitch? To define a more workable agenda, we leave out all the processes at the levels of harmony, melody and rhythm. Leave out the recognition of (i.e. the identification of, and the ability to follow different sounds produced by) the voice and guitar. What are we left with? Segregated, distinct units, each held together internally by the primitive heuristics of auditory perception.

Certainly, such grouping mechanism is much less robust without all higher-level processing we left out. These levels ensure the creation of expectations and direct future searches. While listening to a friend talking, our attention mechanisms will keep irrelevant sound events in the background. But the fundamental capabilities, when working on their own, must be able to initiate the learning processes in the first place. This by no means restricts the architecture to bottom-up functioning. It just limits the model to function at the so-called primitive auditory analysis (i.e. excluding schema-based processes). For fascinating accounts of the ability of lower level perception mechanisms to function while higher cognitive processes don't, see (Sacks 1987, 1995). Subsequently, I tackle what I view as the most fundamental questions of the mechanisms of perception in general, and use the field of auditory perception as the testing grounds for the proposed model, see chapters 7 and 8.

What primitive cues do we use for our heuristics, and how can sounds be processed to reveal such cues? The relevant evidence from physiological and psychoacoustic research is briefly sketched in the following sections.

2.3 Auditory sensory periphery

The cochlea is the main organ of the inner ear. This coiled tube contains the basilar membrane, which runs midway along its length. Eardrum vibrations result in movement of the oval window, which in turn generates a compression sound wave in the cochlear fluid. This compression wave causes a vertical vibration of the basilar membrane (see Figure 2-1). Along the basilar membrane are located nearly 10,000 inner hair cells in a regular geometric pattern. When the basilar membrane vibrates, the stiff hairs protruding from each hair cell deflect, causing a chemical reaction within the cell body which leads to the firing of short electrical pulses in the nerve fibres. These fibres bunch together to form the auditory nerve that connects the sensory input to the following stages of auditory processing in the brain (see Quatieri, 2002).

Different regions along the basilar membrane respond maximally to different frequencies of the input stimulus (see Figure 2-1). The sensory cells along the basilar membrane respond to its vibrations. The cochlea as a whole can be modelled as a filter bank, which, rather like a Fourier transform, converts time signals into their equivalent spectral representation. Still, large differences exist between the standard Fourier transform and the functioning of the basilar membrane. Measurements show a roughly logarithmic increase in bandwidth along the basilar membrane. When modelled as a filter bank, we get filters of approximately constant Q, with an asymmetric frequency response (steeper falloff at high frequencies for each filter). In this sense, the functioning of the basilar membrane can be modelled by a wavelet transform (Quatieri 2002, page 403). Based on physiological data, other models exist, such as the gammatone filterbank for basilar membrane simulation and various computational models for the simulation of cochlear inner hair cells firing patterns.

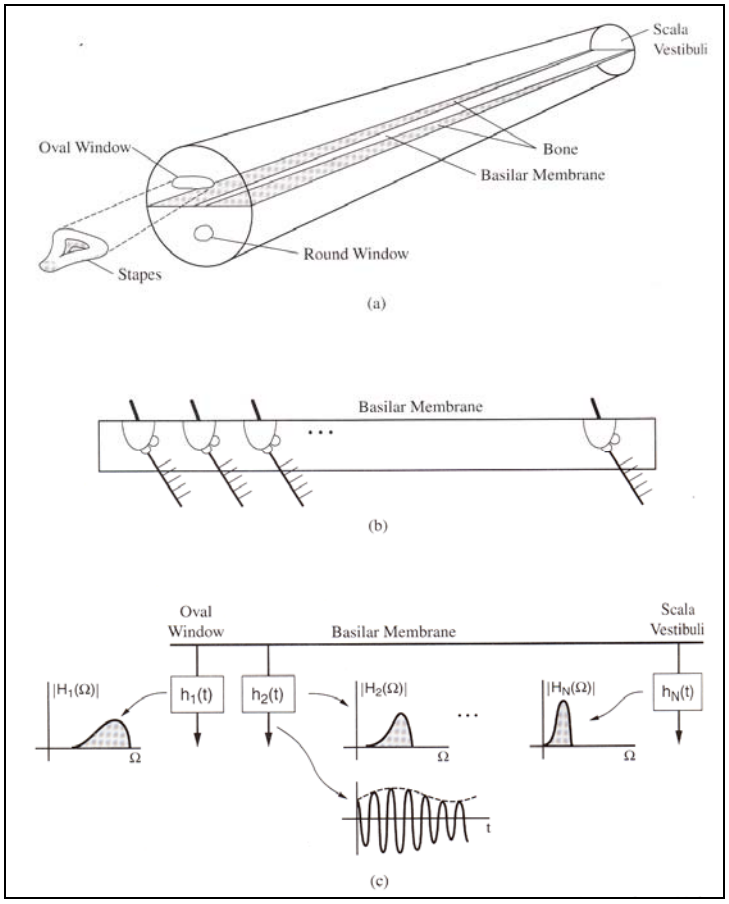

Figure 2-1: Schematic of front-end auditory processing and its model as a wavelet transform: (a) the uncoiled cochlea; (b) the transduction to neural firings of the deflection of hairs that protrude from the inner hair cells along the basilar membrane; (c) a signal processing abstraction of the cochlear filters along the basilar membrane. The filter tuning curves, i.e., frequently responses, are roughly constant-Q with bandwidth decreasing logarithmically from the oval window to the scala vestibule. [From Quatieri 2002, p 402]

This is not the end of the story. Cochlear inner hair cells fire at random at low rates (spontaneous activity) even when presented with no stimulus. The average firing rate of the fibre bundles (corresponding to different locations along the basilar membrane) follows the spectral content of the stimulus (place theory). That is, one can deduce the spectral energy in the region around the characteristic frequency of that fibre from the firing rate of a specific auditory nerve fibre (inferred from the location of its inner hair cell along the basilar membrane) (Ehret 1997). More than one pure tone (sinusoid) can enter the same cochlear 'filter', rendering the expected behaviour (from a signal processing point of view) harder to interpret (or resolve), since the envelope of the filter's input is now determined by the sum of sine waves over the filter's input spectrum. Relative sine wave phases will influence the filter's output. The spectra of neighbouring filters strongly overlap each other. Lower frequencies below 1 kHz will produce nerve firing in sync with stimulus peaks, thereby mapping input frequency and phase code to output time code (temporal theory). In the case of speech sounds, this synchronized output will tend to lock on the centre frequencies of formants, as whole bunches of sine waves in each formant fit into the sensitive spectral range of single cochlear filters. In addition, since the behaviour of each hair cell is largely random, and no one hair cell is able to fire on each phase, the integration of many auditory nerve fibres sharing the same characteristic frequency is required. Since the timing of firing encapsulates so much information about the input sound, it follows that the preservation of phase relations across different filter outputs is of great importance. Following steady stimulus, each hair cell will reach its so-called 'adaptation' phase, in which it is no longer capable to keep on firing at high rates. Adaptation occurs at different time scales.

From a signal-processing perspective, things could look easier had the output of the auditory nerves provided the kind of cues we think we need for the heuristics inferred from psychoacoustic data. It is tempting to think that direct information of each sine wave existing in the input sound stimulus with accurate time and frequency resolution would make the task of producing an auditory perception machine much easier. This, however, may be less desirable than expected. Our heuristics may be wrong. They might be biased towards the kind of heuristic we are used to think of using the familiar paradigms of Boolean logic or computer algorithms. However, it might turn out that quite other cues are extracted from the auditory nerve code before the process of scene analysis starts. Assuming we know the range of cues used by higher processing centres in the brain, we could then investigate the extent to which cochlear coding fits its eventual use. If, for instance, it turns out that the cues used at higher processing centres can be extracted from raw input signals using efficient signal processing techniques it will then seem futile to use cochlear models. It follows, that blindly sticking to the information processing at the neural level only does not guaranty success. One has to identify the information processing even if it takes place at levels, at which neural substrate serves as a general symbol-manipulation system. I come back to this point when discussing the desired properties of models of perception (see Chapters 6 and 7).

Is it possible to find a minimal and sufficient set of cues to allow for auditory scene analysis? Since we deal with a system capable of learning, we cannot rule out the possibility of learning new feature extracting techniques, which renders our hope for finding a sufficient feature-set futile. This, however, does not mean that a basic set of features cannot be found to allow for the most basic mechanisms of perception to take place. One can argue based on neuropsychological evidence[1], that one kind of processing can take place even when other kinds are disabled. The existence of aphasia and amusia suggests that the perception of music and speech are potentially mutually exclusive, at least in the sense that disabling the one may not disable the other. It follows that either the whole processing chain, including primitive segregation mechanisms, leading to the perception of speech is separate from that of music, or that up to a certain point,

[1] These topics go well beyond the scope of this work. For further reading, consult any of the fascinating phenomena of agnosia, aphasia and amusia in cognitive psychology literature and the accounts of Dr. Oliver Sacks (1987, 1995)

they share lower levels of auditory processing. It is, of course, entirely possible that the perception of instrumental music, for instance, makes use of cues which are not needed for the perception of speech and vice versa, but it does make sense that a set of general basic cues are being used for all purposes of grouping sound. How can we then find out what constitutes such a set of general grouping cues? Two very successful techniques that have so far revealed much about the organisation and function of perception mechanisms are described below.

2.4 Psychoacoustics

A group of listeners is presented with an audio fragment for testing some grouping mechanism. For example, (Bregman 1990) tested whether the principle of proximity holds in the auditory domain (see section 4.4 on Gestalt psychology). To eliminate other possible parameters from influencing the results, simple tones having constant frequencies and amplitudes with clear-cut on- and off-sets were used. Using complex tones, for instance (containing multiple simultaneous, possibly harmonically related sinusoids), could introduce other possible grouping pressures between parameters such as amplitudes, spectral shapes, spectral spans, etc. A sequence of alternating pure tones is played at two frequencies Δf Hz. apart, with Δt s. gaps between them. All tones have constant intensities and durations (see Figure 2-2). The tendencies of listeners to hear the tones as if they belong in different groups are recorded while varying Δt. This test clearly shows that shortening the gap between the notes results in an increased tendency for grouping successive tones of the same frequency. Such an experiment suggests the existence of an elementary grouping principle, in this case, based on proximity. Even is such a simple test, it seems that results depend on attention related processes (Bregman 1990, p. 643). How does this grouping principle interact with other heuristics? Is this pressure simple (non-compound) or is it the combined result of elementary processes? How realistic is it to assume the interaction of different heuristics for scene analysis (Bregman 1990, p. 32-33). Such assumptions are more a result of existing paradigms (in this case the world of artificial intelligence) and less from explicit knowledge of the way the brain works. Viewed in this context, there is always the danger of moulding experimental data into an existing computational model. Nevertheless, it is apparent from the vast collection of experimental data that the various grouping mechanisms, whatever they are, work concurrently, and that however ambiguous the input is, perception will form a coherent structure most of the time. Ambiguous cases may lead to alternating interpretations rather than to a single ambiguous interpretation. Apparently, the concurrent existence and integration of multiple grouping pressures is the solution for dealing with non-complete, contradictory evidence. It follows, then, that the ability to integrate multiple grouping pressures is critical for computer models of perception.

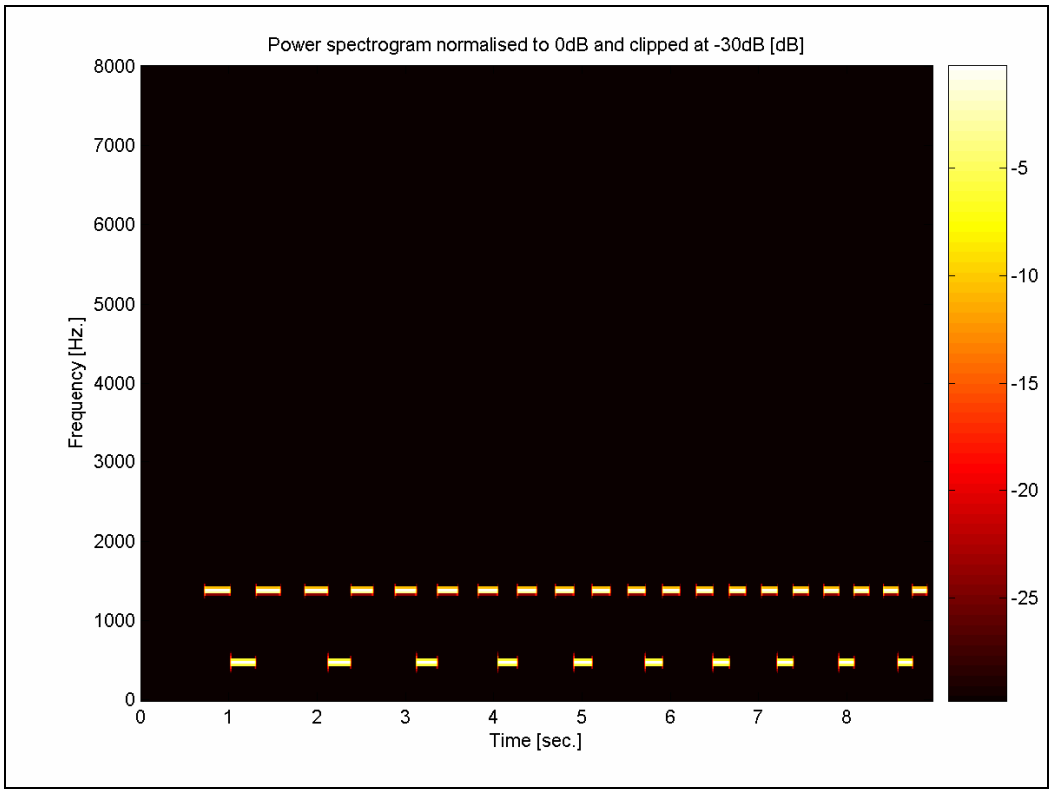

Figure 2-2: Spectrogram of proximity grouping pressures in auditory scene analysis.

In his book, Bregman (1990) briefly touches the issue of the nature of the elementary units of auditory perception. The vast majority of experiments were conducted using basic tone generator units. It is tempting to think that the nature of these tones influences assumptions regarding the nature of the basic units of perception. Bregman himself is aware of this problem and wonders what properties bring about proximity pressures. Is it the proximity between two onsets, or between successive off-and on-sets? Are on- and off-sets considered among different frequencies, or only within iso-frequency channels? Moreover, when one is faced with the complexity of real world sounds one wonders whether sensory input can be arranged into sinusoids in a neat enough way to allow heuristics to work at sinusoid level. In other words, what is the basic level at which elementary grouping principles work?

2.5 Auditory Neuroanatomy

The following brief entry into the field of anatomical investigation of auditory centres is based on Rouiller's 'Functional Organization of the Auditory Pathways' (see Ehret 1997). It is by confronting such data that the reader begins to comprehend how little we actually know about the functioning of the mammalian auditory systems. This information, however, is vital for supporting the main proposal of this work. In what follows, I mainly touch correlation issues between function and anatomical structures.

Neurons of the spiral ganglion contact the hair cells of the cochlea to form the first processing phase of the acoustic stimuli. Their axons form the auditory nerve carrying information to the cochlear nucleus in the brain stem (see Figure 2-3). Two distinct populations of primary auditory neurons exist; for lack of extensive information of cells of the second type, I limit the discussion to type I cells. In the absence of acoustic stimulation, auditory nerve fibres are spontaneously active (inter spike times with Poisson distributions) because of random chemical activity of the inner hair cells. Based on the spontaneous activity of these cells, three principal groups have been described. The first group, which constitutes roughly 60% of the population, has spontaneous activity with a rate above 18 spikes per second, and is very sensitive to acoustic stimulation. This group has a threshold ranging between 0 and 20 dB SPL. Further 30% of the population show spontaneous activity at a rate between 0.5 and 18 spikes per second with thresholds in the range between 40 and 80 dB SPL. Finally, about 10% of the total cell population with a spontaneous activity rate between 0 and 0.5 spikes per second have thresholds of 80 dB SPL and above. Additionally, there is a strong correlation between the three subgroups and their morphological properties. Cells with low and medium spontaneous activity (medium and high thresholds), connect to the inner hair cells on the inner side. In contrast, low-threshold cells with high spontaneous activity contact the inner hair cells on the outer side. Since each individual spiral ganglia cell connects to a single inner hair cell, and due to the longitudinal spectral factoring property of the cochlea, it follows (and experimentally verified), that individual fibres of the auditory nerve are sensitive only to a restricted portion of the frequency range of hearing. The frequency to which the auditory nerve fibre is most sensitive is the *characteristic frequency* (CF). Above threshold, for increasing stimulation intensities, the response area becomes progressively broader. In summary, the principal parameters characterizing an individual auditory nerve fibre are its spontaneous activity, its characteristic frequency and its threshold.

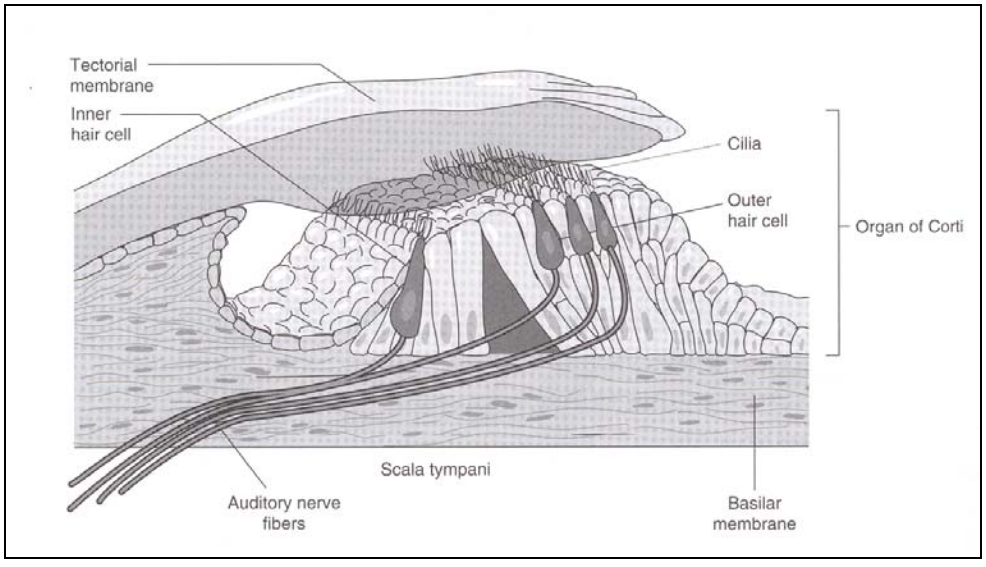

Figure 2-3: A cross section of the organ of Corti, showing how it rests upon the basilar membrane. [From: Goldstein 2002, p. 348]

The discharge patterns in response to white noise or pure tone bursts at CF is the same for all type I auditory nerve fibres. They exclusively show excitatory responses to tone bursts. A brief suppression of activity might be sometimes seen at the offset of a tone before the return to spontaneous activity. Due to cochlear tuning, it is possible to introduce more than a single tone into the sensitivity range of individual auditory fibres. This results in the so-called *two-tone suppression*, which reduces the increase in discharge rate from its normal excitatory rate for pure tone stimuli. This complex response reduces the ability to resolve single frequency components from complex input sounds. On the other hand, it makes it more fit to capture frequency peaks in filtered complex sounds, the so-called *formats*, which are essentially the building blocks of animal calls and human speech. At the low end of the frequency, the response rate of the auditory fibres in comparison with their CF allows them to follow, or react to individual peaks in the input of pure tones. This leads to a temporal synchronised behaviour between fibres sharing the same, or harmonically related characteristic frequencies, called *phase locking*. Phase locking results not only in grouping opportunities based on inter-fibre synchronisation, but also in a far better frequency resolution at the low frequency end as well as the ability to follow amplitude modulation. It is interesting to note that phase locking will track both the peaks of low frequency pure tones as well as the fundamental frequency of complex stimuli (by following periodic temporal peaks) and pulse trains. Phase-locked activity is found in auditory fibres for frequencies below 4-5 kHz. This temporal information is gradually degraded as it ascends the hierarchical processing path to the cortex.

The number of discharges per second is proportional to stimulus intensity above the threshold level at the CF, up to the saturation level of the given fibre. Fibres with lower thresholds will reach saturation for higher intensities while fibres with the same CF, but higher threshold will still function in their proportional range. This apparently enables the system to deal a dynamic range of 120 dB SPL using cells with far lower dynamic ranges. Additionally, there is ample evidence to show that though feedback mechanisms reaching the cochlea, both the tuning curves and dynamic ranges are significantly improved. Additional feedback mechanisms exist, but due to their complex effects, their function is less clear. However, from a control-engineering point of view, the performance enhancement of sensory feedback systems by the dynamic use of just a few feedback mechanisms (if kept stable) can be immense.

The cochlear nucleus is the first nucleus of the central auditory pathways. It contains the secondary auditory neurons that receive ascending information from the auditory fibres, all of which terminate in the cochlear nucleus. Since this is a mere summary of facts, it will suffice to report that the cochlear nucleus is globally divided into 6 major areas based on the morphology of neurons which reside within. Rouiller (Ehret 1997) lists 8 distinct types of neurons, each having typical shapes, sizes, connectivity, and responses. The various spontaneous activity rates are not preserved in the cochlear nucleus. Here typical rates of about 110-120 spikes per second are valid across the population. Some neurons show an equivalent response histogram to that of the auditory fibres. These are called primary-like response types, and have a large peak at the onset of input stimuli followed by a lower-level response throughout the steady state portion of the stimuli. Variations of this response exist which contain a notch between the onset and the steady state sections, the so called on-I type, which essentially behave as pure onset detectors, and fire at rates above spontaneous activity only at onsets. Notice, however, that the association of such detectors with the term *onset* might have to do with the type of stimulus used in the experiments (specifically, the use of short pure tone stimuli with no amplitude modulation, and no dynamic behaviour will result in the firing of such an 'onset detector' only at stimulus onsets). Other response types appear for the first time in the cochlear nucleus. The chopper response shares its envelope with the primary-like response types, with the additional feature of having multiples peaks separates by regular time intervals that are not related to the frequency of the simulation tone. The use the brain makes this type of response is not yet clear. Since time intervals decrease with an increase of stimulus intensity, it has been suggested that chopper response types could serve as intensity to rate coders. In addition, inhibitory connections are

found mainly in the dorsal cochlear nucleus, and result in complex response patterns, showing significantly different behaviour in anesthetized animals as compared to non-anesthetized ones. Response types which make use of inhibition include the *pauser* (suppressed within some time interval following onset) and the *buildup* (showing a progressive increase in discharge rate for the duration of the stimulus). Further, these cells use inhibitory connections to suppress neighbouring characteristic frequencies (See Figure 2-4).

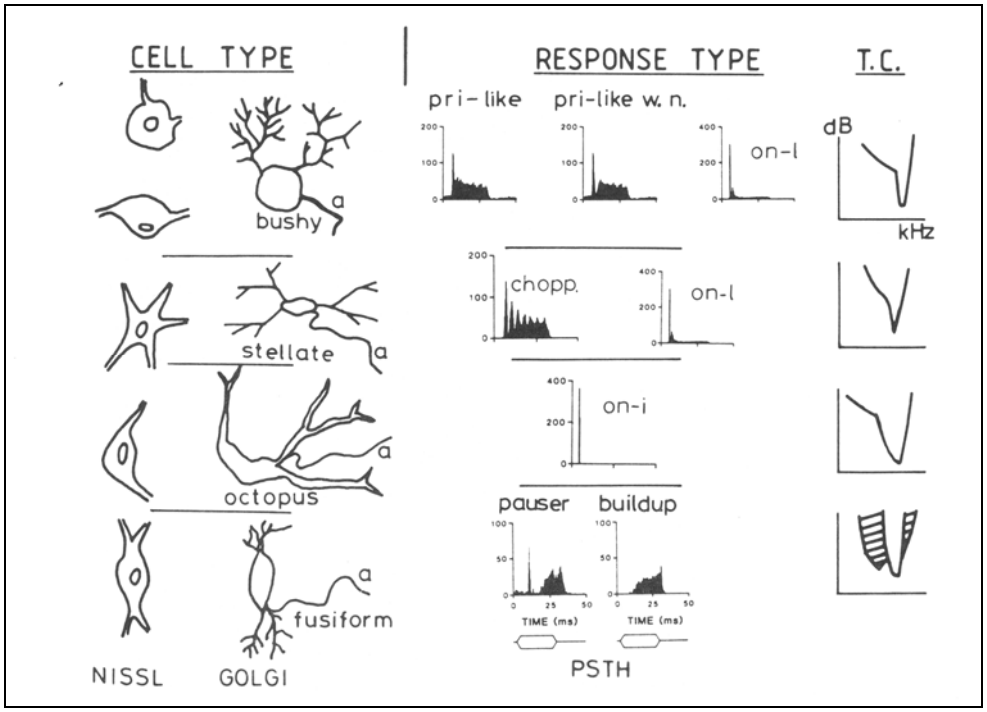

Figure 2-4: Correlation between cell types in the cochlear nucleus (as they appear in Nissl or Golgi material) with single unit response types (indicated by their peristimulus time histogram [PSTH]), as well as with their tuning curves (T.C). The duration of the tone bursts was 25ms, and they were presented 600 times for each PSTH computation. For tuning curves, the dashed portions indicate zones of inhibition, while the white inner portion of the curve represents the excitatory area, as compared with the spontaneous discharge rate. a = axon; on-i = discharge only to stimulus onset; on-l = onset response and weak discharge during rest of tone-burst; pri-like = primary-like; pri-like w.n. = primary-like with notch. chopp. = chopper. [From Ehret 1997, p. 18].

A near strict tonotopic spatial organisation following the CF of incoming auditory fibres exists throughout the cochlear nucleus as well as in most auditory nuclei (see Figure 2-5). In addition, the different zones within the cochlear nucleus maintain parallel connections with the incoming fibres, the exact arrangement of which depend on the type and place of each auditory fibre. The cochlear nucleus receives the information from the auditory nerve fibres, processes it, and distributes the modified signals to higher auditory nuclei. This distribution pattern is 'far from understood in all its detail because the high degree of collateralization of the axons from the CN makes this pattern very complex' (Ehret 1997, p. 25). In any rate, it is clear that from here, the picture becomes much more complex, and that multiple, parallel paths exist among auditory nuclei. Likewise, paths can be found which provide reciprocal connectivity (feedback) between the various nuclei.

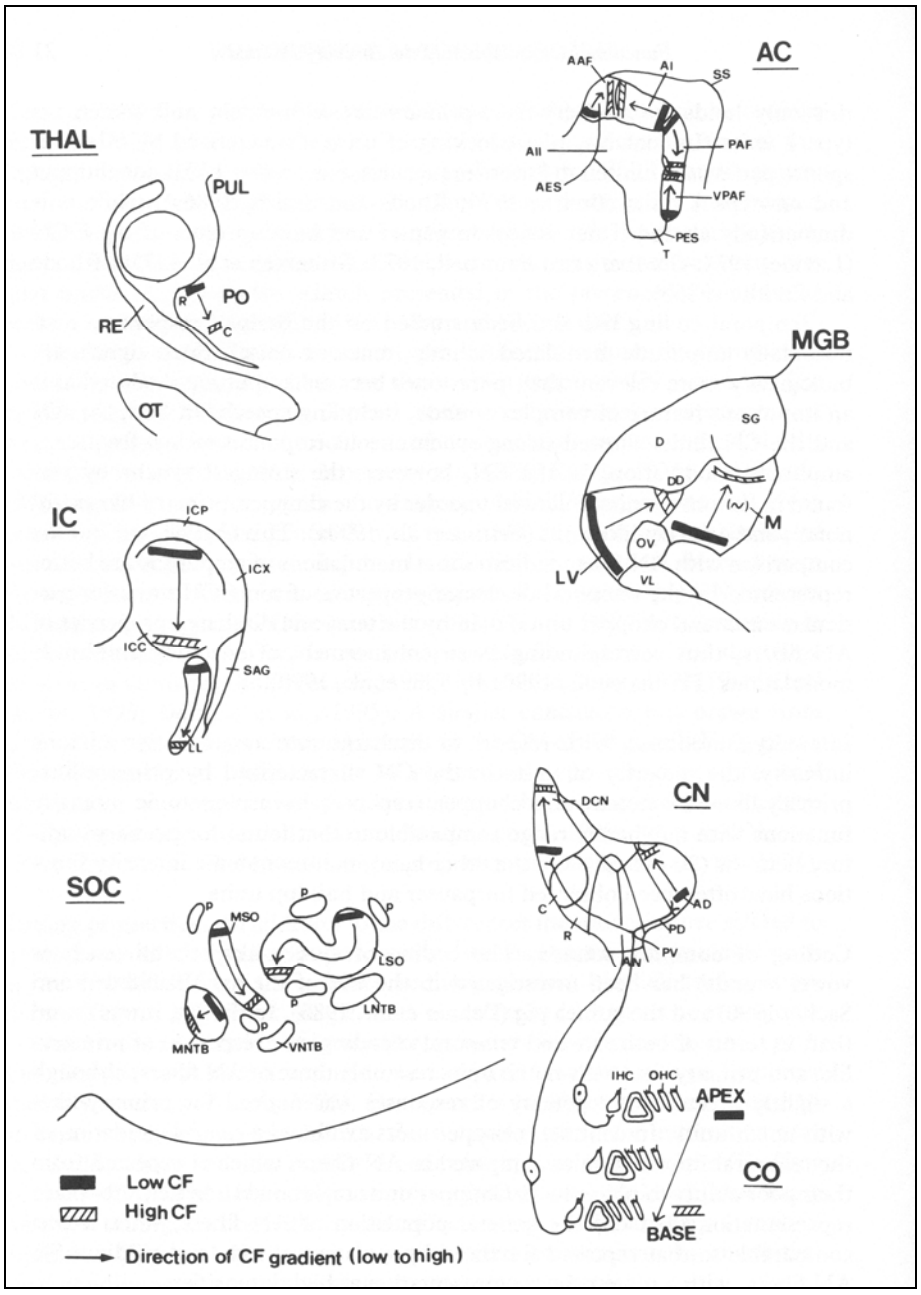

Figure 2-5: Schematic representation of the various auditory nuclei (in the cat) from the cochlear nucleus to the auditory cortex, with illustration of their respective tonotopic organization (preferential spatial arrangement of the neurons with respect to their CF). The arrows indicate the direction of the increasing CF gradients. [From Ehret 1997].

Less data exists about the function and anatomy of the *Superior Olivary Complex* (SOC). In the *medial nucleus of the trapezoid body*, one of the nuclei of the SOC, there is evidence for a 'primary-like with notch' response-type with narrow tuning curves. In the *lateral superior Olivary nucleus* (LSO), tonal selectivity is good and most of the cells show the chopper response type. Throughout the SOC there is a slight degradation of temporal coding from the cochlear nucleus, and therefore, of phase-locked activity. The latter is now restricted to the range below 2-3 kHz. The LSO, though arranged tonotopically, contains the range of higher frequencies than the tonotopically arranged MSO (*medial superior olivary nucleus*). Species with sensitivity to high frequencies, such as mice, moles, dolphins and bats tend to have a large LSO and a small (if not absent) MSO. Humans, in contrast, as well as other species with sensitivity at low frequencies have a large MSO, and a smaller LSO. The SOC is the first location in the auditory processing chain which combines the information from both ears. From here onwards, we witness all the possible combinations laid in spatial maps. Some locations combine the information from both ears as excitatory input, while others show excitatory response to stimuli from one ear, while being suppressed by information from the other. The arrangement is symmetric so that the right hemisphere SOC mirrors the left. The integration of information from both ears entails the processing of both spectral and temporal information. From here on, I leave out all issues regarding the processing of binaural information for the following reasons:

1. It is possible to achieve most feats of auditory perception with the exclusion of some localisation functions which depend on binaural information with the use of one ear.

2. Excluding binaural integration will reduce the model's complexity.

3. If the model handles the integration of cues from different modalities then the later integration of binaural processing is possible.

According to Ehret (1997), *'the number of centers of the ascending auditory pathway up the midbrain seems to have remained rather constant in the phylogeny of vertebrates. Input nuclei homologous with the cochlear nucleus complex, second and third order nuclei in the superior olive complex, and nuclei of the lateral lemniscus are present in amphibians. In these vertebrates, the forebrain is much less developed compared with that of mammals, and an auditory cortex does not yet exist, so that sound analyses in the highly vocal anurans must be accomplished at the level of the midbrain and thalamus. From an evolutionary point of view, therefore, we can expect the midbrain to be essential not only for all sorts of acoustical information processing in the spectral, temporal, and spatial domain but also for the generation of response behaviour to sound'.*

The midbrain therefore already contains the main processing functions which allow auditory scene analysis. It may be, however (Ehret 1997, p. 304), that, *'compared to amphibians, the auditory midbrain of mammals … has lost much of its direct influence on auditory perception and behaviour…'* It is my belief that before embarking on higher-level capabilities such as music and speech recognition, a working solution for the problem of 'primitive' auditory scene analysis should be found. Figure 2-6 shows a diagram of the main pathways of the ascending auditory system starting with the left cochlea. The diagram is taken unmodified from (Ehret 1997, p. 261). According to the Ehret (1997), we should be able to find neurological evidence for capabilities of primitive auditory scene analysis. Due to the complexity of connectivity of both outputs and inputs it is conceivable that the behaviour of single neurons will show much more complex patterns than what we saw in the auditory nerve fibres, or even in the superior olivary complex. The existence of the feedback loops (both with higher and lower-processing centres) can also be taken as evidence for the capabilities of the inferior colliculus (IC).

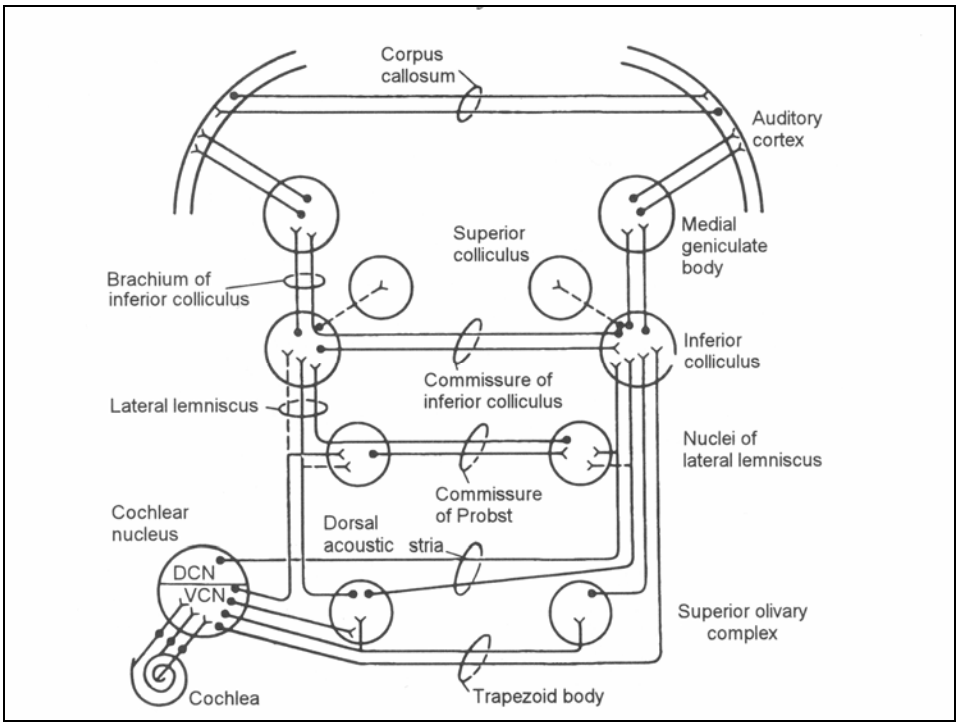

Figure 2-6: Diagram of the main pathways of the ascending auditory system starting with the left cochlea. The same connections would have to be drawn for the other side. DCN = dorsal cochlear nucleus; VCN = ventral cochlear nucleus. [From Ehret 1997]

As it turns out, there seems to be a remarkable level of complexity in the response of single cells to auditory stimulus. Ehret (1997, p. 269) lists the properties of individual cells in which complexity is found at this stage:

1. shape of the excitatory tuning curve that demarcates the frequency response area within which a neuron can be excited above its spontaneous response rate by a pure tone;

2. presence and shape of inhibitory tuning curves that demarcate areas of inhibition within which sound inhibits the excitatory response to a certain extent;

3. presence and shape of facilitatory frequency areas within which sound facilitates or enhances the response in the excitatory response area without being excitatory by itself;

4. spontaneous activity;

5. response latencies;

6. response patterns;

7. shape of rate-intensity functions;

8. properties of spectral integration;

9. properties of temporal integration;

10. responsiveness to modulations in frequency (FM) and intensity (AM);

11. responsiveness to binaural interactions and auditory space cues;

12. responsiveness to other modalities besides sound;

13. sensitivity to influences of state dependent factors of the animal (arousal, attention, learning, anaesthesia, etc.)

The behaviour of auditory nerve fivers is far simpler than the list above. Moreover, the response predictability at the early stages of the auditory path gives way to complex, dynamic responses. As Ehret puts it, *'In addition to complex excitatory and inhibitory response areas, facilitation and occurrence of off-responses indicate that simple relationships between sound stimulation and neuronal response behaviour in the ICC* [central nucleus of the inferior colliculus] *cannot be expected, not even for single-tone stimuli'*. See Figure 2-7 (from Ehret 1997, p. 270) for two tuning curves of individual ICC cells. Response latencies exist in the range of 4 ms (direct connections from cochlear nucleus) for high intensity stimuli through more typical 10-13 ms for the maximum number of neurons within the various pathways from the auditory nerve to the IC. About 10% of the ICC neurons in the cat show latencies of more than 20 ms with a maximum of 50 ms; this could only be explained by the affect of inhibitory connections. Responses include (see Figure 2-8) the primary-like, tonic and phasic-tonic responses (a-d), onset (phasic) responses (e, f), pausers with an onset peak followed by a silent interval and a tonic discharge (g, h), long-latency or buildup responses (i, k), inhibition of spontaneous activity (l), inhibition of spontaneous activity followed by an off-response (m) , on-off responses (n, o), phase-locked tonic responses to either AM or FM (p), and onset and tonic-off responses with chopper characteristics (q). Offset responses occur mainly at higher sound intensities and with complex sound stimuli. However, response patterns in the IC are not fixed for a given cell. They are dependent on the intensity and type of the sound stimulus. The relation between intensity and rate of discharge is no longer a monotonic function. Discharge rates can fall after reaching certain intensity levels tuning cells in both the frequency and the intensity. Moreover, discharge rates have been shown to depend on the frequency spectrum and the temporal characteristics of stimuli. Ehret concludes that the available data on response patterns and rate-intensity functions demonstrate that *'the message of single neurons of the ICC, transferred by their discharge patterns and rates to higher auditory centers, cannot simply be related to a certain physical aspect of the sound stimulus. Discharge patterns and rates may convey information about sound intensity, stimulation frequency, complexity of the stimulus in the spectral and temporal domain, and sound direction all at the same time… how order can be introduced into a seemingly disastrous diversity, is still a secret of the nervous system'*.

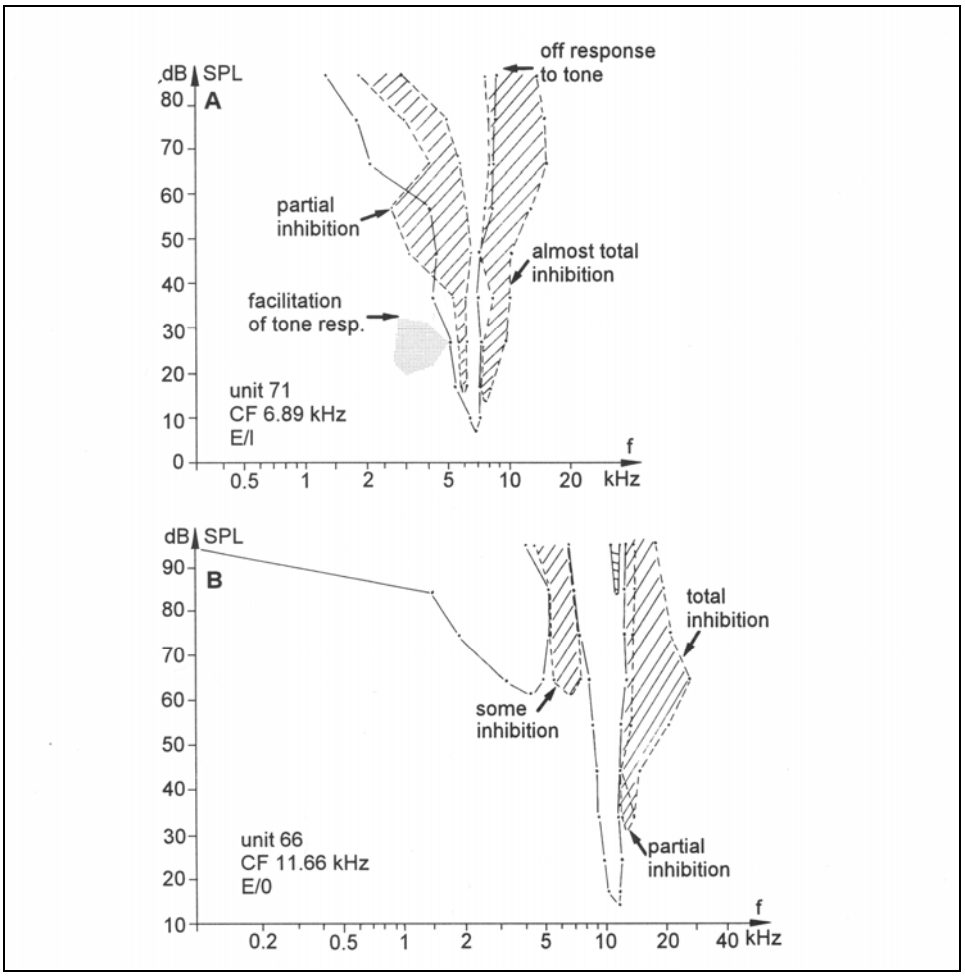

Figure 2-7: Two neurons of the central nucleus of the inferior colliculus of the cat with excitatory tuning curves and inhibitory and facilitatory areas. Characteristic frequencies (CFs), binaural response types (e.g., E/I), strengths of inhibition of a CF-tone response by another tone within the inhibitory area, and the occurrence of off-responses are indicated. [From Ehret 1997]

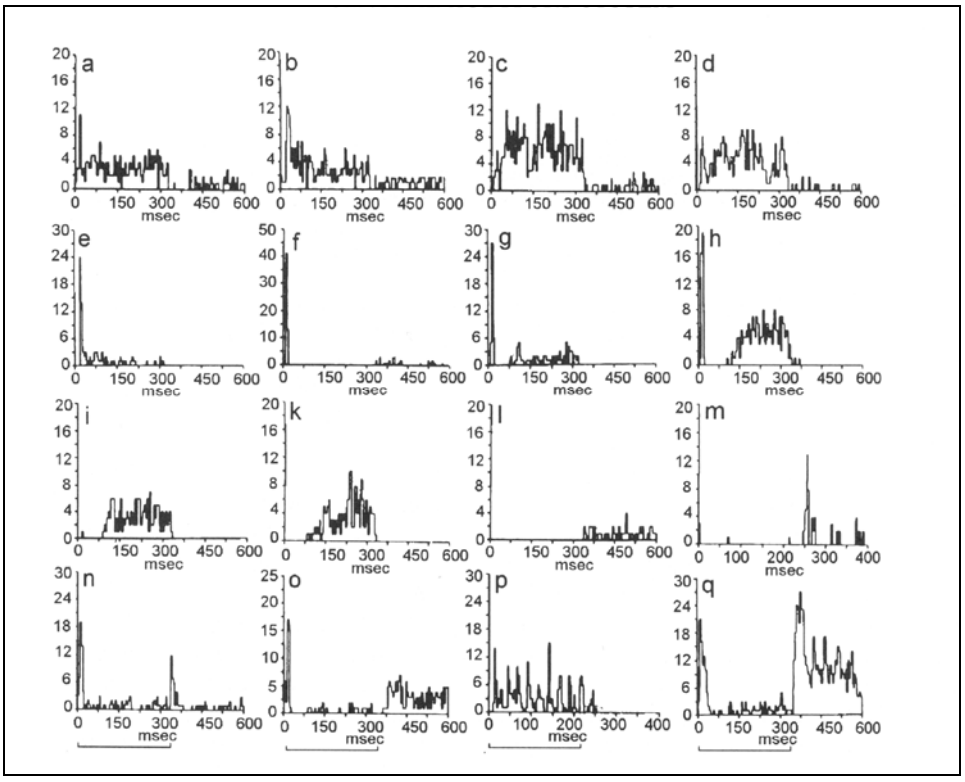

Figure 2-8: Peristimulus time histograms showing basic response patterns of ICC neurons to singe-tone stimuli (a-o), two phase-locked tones (p), and tone bursts in a background of continuous broadband noise (q). The lines underneath the lowest histograms indicate the length of the tone stimulus. The responses of 20 stimuli are summed up. a, b: phasic-tonic (primary-like) response; c, d: tonic response; e, f: phasic response; g, h: pauser type; i, k: long latency (buildup response); l: spontaneous activity inhibited; m: off-response with chopper characteristic; n: phasic on-off response; o: phasic on-response with tonic off-response; p: tonic response to amplitude modulation (AM) to the two-component phase-locked stimulus (610 + 660 Hz) with envelope following of the 40 Hz modulations; q: on-response with chopper response to noise after tone-off. [From Ehret 1997, p. 274]

The complex combination of excitatory, inhibitory and possibly facilitatory areas, with dependences on intensities bandwidths and spectral shapes makes predictions of response type and discharge rate very difficult to make. However, there is evidence for the existence of *tuning curves* for the analysis of complex sounds. These can be produced by integration of stimulus energy within the curve's frequency sensitivity zone. Sound outside this area will not affect discharge rate. These bandwidths of spectral integration are termed critical bands. Critical bands off about 60% of the ICC population of the cat remain rather constant in width for different stimulus intensities. This differs greatly from the behaviour of tuning curves. It has been estimated that there could exist some 20 non-overlapping zones of spectral integration, or critical bands in the cat. Figure 2-9 depicts some tuning curves. Further study of responses to AM and FM stimuli within the critical band reveals capabilities to track modulation rates (See Figure 2-10). Critical bandwidths increase with increasing characteristic frequencies of neurons in the ICC. They are roughly constant for a given CF and independent of sound intensity. The ICC as a whole is highly organised into iso-frequency planes. This arrangement seems to be more flexible, allowing more neural processing tissue for frequency ranges which demand more attention, such as echolocation frequencies in the bat. Neurons with different latencies are orderly mapped within iso-frequency planes create iso-latency lines. In addition, arrangements have been found

within iso-frequency planes, which correlate with both response thresholds and tuning sharpness characteristics. In general, according to Ehret, as far as single tones are concerned, *'stimulation with a certain frequency leads to a high responsiveness in the center of the respective iso-frequency plane but to a central cylinder of silence in other iso-frequency planes. The pattern of spread of excitation from the tone-related iso-frequency plane to other planes depends only on the on the frequency and intensity of the stimulus tone'*. Whether such predictability exists for complex auditory stimulus is not yet clear. Additional maps have been located, which order responses to modulation frequencies, and azimuth angles.

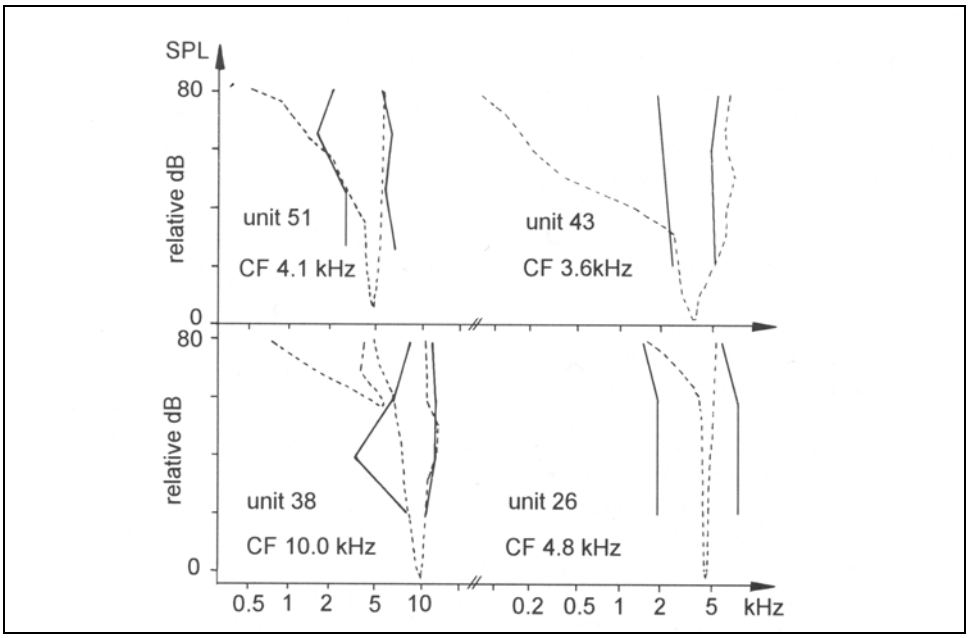

Figure 2-9: Examples of tuning curves in response to complex sounds (tone bursts in critical bandwide noise; heavy lines) compared with single-tone excitatory tuning curves (dashed lines) of the same neurons from the ICC of the cat. [From Ehret 1997, p. 278]

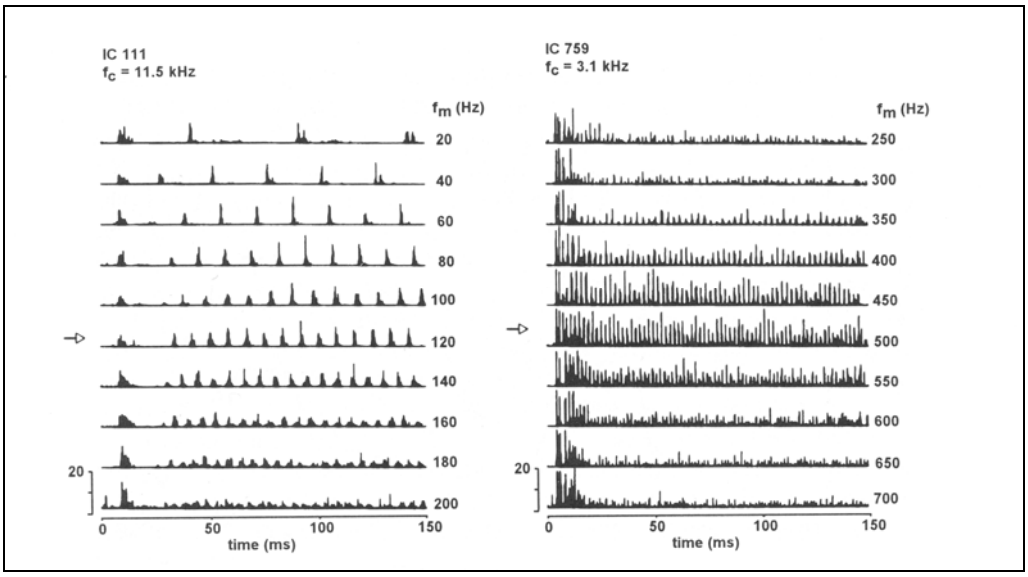

Figure 2-10: Responses of a single neuron (left) and a small group of neurons (right) of the ICC of the cat to 100% amplitude modulations (AM) of the CF tone with various modulation frequencies (f_m). The PST histograms are summed over 100 (left) or 30 (right) repetitions. The AM-following response is very clear. Arrows indicate the best modulation frequencies by a rate criterion. F_c = characteristic frequency. [From Ehret 1997, p. 279]

It follows that mapping within isofrequency planes does not follow independent axial geometries only, but also such covarying topologies as concentric zones. This opens the way, theoretically, for many more mapped structures to coexist in the ICC. Moreover, though not proven, Ehret argues the possibility that single neurons could in fact contribute to several maps at the same time, where each map selectively uses a different property of the neuron's response. The complexity is such, that any analysis regarding the function and utility of such maps, at least at this point, will be very speculative. The standard interpretation is that these maps function as primitive feature detection maps for the use of feature *combination* maps such as those found in the auditory cortex of the bat. From this point of view, the ICC integrates, organizes and encodes all the information from all other lower centres of auditory processing into 'spatial and temporal patterns of excitation, the readout of which can be "understood" in higher centres' (Ehret 1997, p. 300). Though this description is plausible, it leaves two critical questions unanswered:

1. What is the function of the feedback connections from those very same centres which are supposed to read the spatio-temporal patterns? If such function consists of the regulation or adjustment by the reader of the patterns to be read than the above hypothesis fails by drawing the line between the passive picture and the active reader.

2. Even if the dynamic patterns encode the set of features to be read by higher centres, then this picture remains as complex as it was at the beginning. In other words, this hypothesis helps little in clarifying how such mapping can be used for auditory scene analysis. 'Reading' the patterns does not seem easier than 'reading' directly from lower auditory centres. There is, however, another interpretation which, while not contradicting the physiological data in any way, could allow us to understand this complexity in a more abstract way.

The medial geniculate body (MGB), together with the lateral part of the posterior complex of the thalamus (PO) and the posterolateral sector of the reticular nucleus of the thalamus (NRT) form the auditory structures in the thalamus. Of the entire MGB population, only about 3% show the ability to lock onto phase. The great majority thus takes part in following transient events. Four cell classes have been distinguished having typical response patterns. Apart from about 15% which show no response at all to click train stimuli, the so-called *lockers* form about 60% of the lateral part of the NGB (LV), and synchronise to individual clicks up to a limiting frequency of clicks per second. The *groupers* make 10% of the LV, and respond only in between clicks being probably suppressed by the clicks themselves. The *special responders* occupy some 15% of the LV, and tend to handle the click group as a whole, indicating on- or off-sets for a limited range of repetition frequencies. Some lockers have been found which respond better for a selective range of repetition frequencies and could act as detectors for such stimuli (see Figure 2-11). Apart from the usual existence of iso-frequency planes, there is strong evidence to suggest the consistent mapping within each plane of different response types and sensitivities. It has been statistically shown that cells responding to the stimulus of pure tones, but not to broadband stimuli are not situated in the same areas occupied by other cell types. The same applies for the different response types (see Ehret 1997, ch. 5).

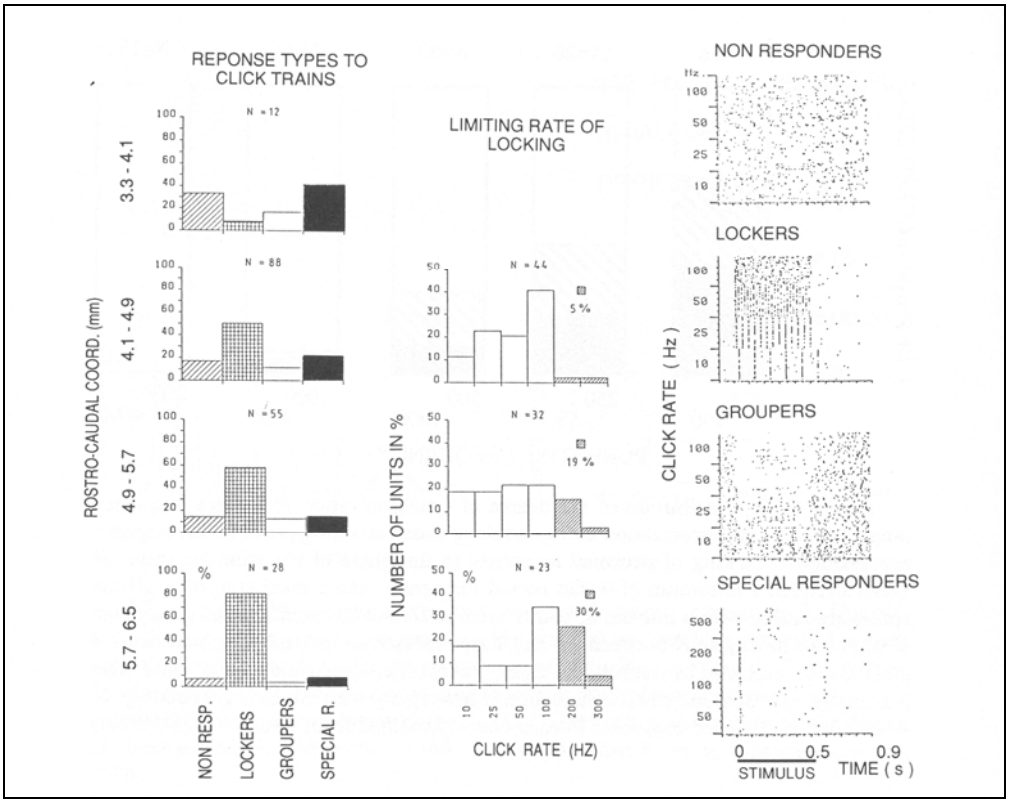

Figure 2-11: Distribution of the coding properties to click trains in LV as a function of the rostrocaudal coordinate. [From Ehret 1997, p. 336]

Generally, the tuning curves in the medial division of the MGB (M) are broader and often show multiple peaks. *'The responses are more labile … and often disappear when repetition rate exceeds one per second. A tonotopy is not evident. The units respond as well to simple acoustical stimuli as they do to complex ones (i.e. cat calls)…the receptive fields of neurons in the M can be modified by learning… this functional plasticity is more prominent and long-lasting than the one observed in the vertral division'* (Ehret 1997, p. 343). The dorsal division (D) shows no signs of tonotopic organization, and forms the most complex section of the MGB. A large proportion of D contains units which do not respond to simple acoustic stimuli, or quick habituation units. *'In the awake cat, one-third of the units are unresponsive to simple stimuli but respond to complex sounds like cries of kittens … The D … may be involved in early identification of familiar acoustic stimuli that are of fundamental importance for a given species'.*

The same response types as those at the thalamic level are found in the auditory cortex, but with an even greater diversity (see

Figure 2-12). The transient on responses are most frequent. They vary from cell to cell in their latency, temporal precision and number of evoked discharges. Sustained and offset responses represent some 10% and 30% of the responses respectively. Responses, however, of a given unit are *'not rigidly fixed but can change as a function of intensity, spectral content, or localization cues of the acoustical stimulus'*. Three principal tuning curves have been found in the cortex: a narrow type (with possible inhibitory connections from neighbouring frequencies), a broad type (occupying about one third of the dorsal part and having reduced latencies and thresholds for two-tone combinations), and a multiple-peak type (a possible convergence of multiple narrowly tuned units). Both monotonic and non-monotonic rate functions have been found. The best intensities of the non-monotonic units seem to be well distributed in a 50 dB range, possibly forming an amplitude map for a given frequency. Such findings were found in the moustached bat at the frequency of the second harmonic of its echolocation pulses. Since the majority of units respond only to transient cues, their intensity coding will shift due to of background noise and will code the dynamic changes above that level. Responses to amplitude modulations have revealed units which respond either to ascending or descending phases of the modulated stimulus, but not to both. Units show best modulation frequencies and complex dependencies on intensity.

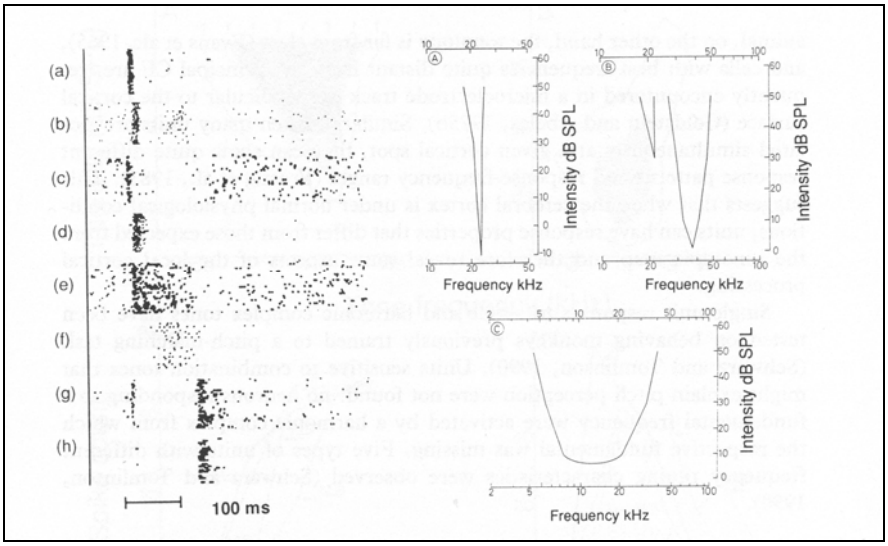

Figure 2-12: Responses of cortical units to tone bursts. Left: Response patterns to a 100 ms tone burst indicated by the calibration bar. Transient ON responses with different duration and latencies (a-d), sustained responses (e and f), transient ON and OFF (g) and pure OFF responses (h). Dot displays represent for each unit the response to 25 consecutive stimulus presentations. Right: examples of three types of tuning curves, typical of the primary auditory cortex: narrowly tuned unit (A), multiple peak unit (B), and broadly tuned unit (C). [From Ehret 1997, p. 359]

Although the evidence for the existence of units with specificity for a given set of natural sounds such as animal calls is difficult to interpret, some data clearly shows complex sound detection behaviour. Units have been found which respond to a few complex sounds, but not to simple tones. Moreover, their response is persistent over a range of frequencies even if the stimulus spectrum lies outside their expected excitatory tuning curve. While most units will respond to more than half of the given stimulus repertoire, there seems to be an agreement among researchers that neurons at the cortical level need some particular acoustical features following particular time sequences in order to become fully activated. These units also respond more vigorously to natural animal calls than to pure tones. Experiments have been done, in which recorded such calls were reversed in time, thereby keeping their instantaneous frequency unaltered. This resulted either in the total abolishment of response, the reversal of the temporal response pattern, or in other 'unrelated' responses. These results could have been anticipated due to the interactions of units with different latencies, implying a strong dependence on temporal structures, or as de Ribaupierre puts it,

'The response to an acoustic element can be significantly influenced by the preceding component and cortical neurons do not act as passive filters. From all these observations one can say that an auditory cortical unit can be strongly activated by natural stimuli but responds rarely in a selective way to a unique vocalization taken from a set of many stimuli. Even in the latter case, the neuron does not recognize the call as an entity, but responds to some of the acoustical features present in the time course of the call or to single call elements.' (Ehret 1997, p. 371)

3 Computational Auditory Scene Analysis

Algorithmic Models of Auditory Perception

The field of computational auditory scene analysis (CASA) differs from earlier computational mechanisms for automatic speech recognition (ASR) and sound processing in that it accepts, at least in principle, the functional approach to the problem of perception as abstracted from Marr's 'Computational theory of vision' (1982) in general, and proposed in Bregman's 'Auditory scene Analysis' (1990). As Rosenthal & Okuno put it (1998, preface p.XI), 'Researchers attempting to build computational models of hearing have concluded that systems which process information at strictly ordered levels of abstraction lack the flexibility to deal with real-world sound. Researchers who are primarily interested in architectural considerations, for their part, have seen CASA as a fertile proving ground for their ideas. At the same time, the increasing desirability of structured electronic archives, more accurate ASR, and more sophisticated hearing aids have provided motivation and opportunities from the applications viewpoint'. In this chapter, I review a selection of such models which are relevant to the discussion of the present work. Consequently, signal processing models of the sensory periphery as well as schema-driven models (e.g. music analysis and speech recognition) have been left out.

In section 3.1 I review several computational models of elementary auditory units, and the primitive grouping thereof while in section 3.2, I describe a few more generic architectures designed to tackle the problems of heuristic interaction and competition on one hand, and the inclusion of top down processing on the other. For each model, I point out the main ***Preprocessing techniques*** *used (e.g. the processing applied to audio signals before arriving at the actual model), the* ***Features used*** *by the model for scene analysis, the explicit (i.e. declared by the authors), the* ***Assumptions*** *used for the design of the model, and the* ***Goals*** *the model had to reach (or the main tasks which the model was supposed to be capable of). In addition, I list* ***Emergent psychophysical properties*** *in cases where the model shows behaviours which was not explicitly coded for, and are in line with psychophysical evidence. Such emergent behaviour is of great value for the evaluation of psychophysical plausibility and of future prospects of model extension.*

3.1 Feature extraction and primitive scene analysis

Models for the identification of the elementary units of perception and for the primitive grouping of such units.

3.1.1 Fishbach, A., Auditory Scene Analysis: Primary Segmentation and Feature Estimation [1]

- **Preprocessing:** Short time Fourier transform
- **Features used:** Temporal and spectral intensity- and phase-discontinuities
- **Assumptions:** equivalence of discontinuities and event properties (such as the assumed equivalence between temporal discontinuities and onsets). Legitimacy of the usage of non physiological models for the extraction of psychoacoustic features. In this case, the use of the complex spectrogram
- **Goals:** Segmentation into the proposed elementary units of perception

Fishbach (1994, and Rosenthal & Okuno 1998, chapter. 15) proposes a model for the estimation of elementary features, assumed essential for scene analysis. The motivation for the proposed model is the lack of common definition of the elementary units of auditory perception. Following psychophysical evidence for sensitivity to discontinuities in input stimuli, Fishbach works out a mechanism for the detection and primitive grouping of bordering lines based on intensity and phase discontinuities in both temporal and spectral directions. Fishbach explicitly assumes the tendency of perceptual mechanisms for grouping rather than segregating all locations which lie within computed border-lines. He proposes that as long as the main grouping phenomena of auditory scene analysis are not hindered by the choice of preprocessing, that is, as long as the correct features are extracted from input stimuli, it is not strictly necessary to match the filtering characteristics of the inner ear. As I argue later on, I regard the gradual detachment of perceptual features from physiological substrates not only acceptable, but also principally motivated by psychophysical evidence.

Large values of directional derivatives of phase and absolute values of intensity and phase discontinuities in time and across frequencies are computed from the complex spectrogram. Parameters for the estimation of on- and offsets were derived from the interpretation of psychoacoustic data relating derivative values to the perception of onset events. This leads to four binary boundary maps, which are then merged into a single map that defines the borders between units in the spectrogram. A recursive segmentation algorithm groups spectrogram pixels into such units. These units are assumed to serve as the elementary units of perception for to be used by a separate scene analysis engine. For the examination of the model, 'The segmentation of auditory scenes which contain more than one source was tested by means of resynthesis of manually selected segment groups'. Based on this test, Fishbach suggests that the model can serve as a basis for computational auditory scene analysis grouping models.

[1] Computational Auditory Scene Analysis (Rosenthal & Okuno 1998), chapter. 15

It is interesting to note the usage of terms such as 'onset' in computational models of perception. It is my view that such feature-terminology (which is the general rule in the field) confuses signal features with the properties of an event (in this case, onset, that is, the beginning of an event such as a tone or an utterance). Signal discontinuity features in input-stimulus are used by auditory perception mechanisms only as *cues* for the existence of *salient* locations in input stimuli, as should be clear from psychoacoustic data in general, and from the nature of many auditory paradoxes in particular. Such discontinuities still have to be interpreted in the *context* of other cues (see chapter 7). It might be that the general tendency of researchers to 'hardwire' discontinuities to onsets is influenced by the nature of sound fragments used in most psychoacoustic experiments. Such fragments often contain only 'well defined' synthesised tones with unambiguous onsets, that is, with agreement between temporal discontinuities and onsets. From this perspective, it is not entirely clear to what extent an automatic segmentation phase prior to the consideration of context would contribute to scene analysis. In other words, since the integration with other cues which may be extracted from the input signal seems a must for the proper function of scene analysis, is it advantageous to segment the scene based only on discontinuities (corresponding to a single-threshold policy) prior to the interaction with other cues? There is still not enough evidence to point one way or the other. Moreover, the manual grouping of a subset of segmented elementary units for resynthesis says little of the significance of the choice of unit segmentation itself, other than suggesting that border parameters were not overestimated, that is, that enough borders exist to ensure the safe segregation of distinct events. This does point to the possible usage of discontinuity features (without the automatic segmentation) as (part of) a front-end for a scene analysis engine. There remains the question of the possible need for top-down processing as, evidenced by feedback connections throughout the auditory pathways.

3.1.2 Meddis R. & O'Mard L., Psychophysically Faithful Methods for Extracting Pitch [1]

- **Preprocessing:** Filter bank, inner hair cell model, cross-channel summation of autocorrelation
- **Features used:** Onsets, Pitch
- **Assumptions:** spectral content differs between fore- and background sounds
- **Goals:** segregation of simultaneous vowels
- **Emergent Psychophysical properties**: pitch of missing fundamental, ambiguous pitch, repetition pitch and other pitch related behaviour.

[1] In Computational Auditory Scene Analysis (Rosenthal & Okuno 1998), chapter. 4

Meddis & O'Mard (Rosenthal & Okuno 1998, chapter. 4) implicitly assume the plausibility of pitch extraction at early stages of auditory perception. This follows from their attempt to extract the generally regarded emergent property of pitch by a direct application of autocorrelation followed by cross channel summation on the output of an inner hair cell model. The extraction of pitch by using auto correlation methods reduces the problem to finding the global correlates common to different channels. Following the explicit assumption that foreground sounds differ from background ones in their spectral content (which, since it *might* be the case in some instances, presents one of many possible alternatives for a parallel processing system), it seems reasonable to believe that segregation based on correlation among different spectral channels is possible. The researchers turn to correlation methods in an attempt to get around the problem of recognising the output of high and low channels as belonging to the same object; this while low frequency channels tend to follow the phase of harmonic stimuli and high frequency channels tend to produce beating at the frequency of the fundamental. Low frequency channels can resolve single harmonic component of the complex signal, while high frequency channels cannot. This results in the summation of harmonics at the output. Though such a strategy is straight forward, there is neither psychophysical nor neurological data to suggest pitch extraction capabilities at the level of auditory nerve-fibres.

The algorithm is made up of four stages. A bank of band pass filters is followed by a 'bank of inner hair cell models that provide the input to an autocorrelation stage. The process is completed by a summation process which adds together all the autocorrelation functions. The pitch of the signal is estimated using the reciprocal of the lag of the highest peak in the summery autocorrelation function'. The inner hair cell model suppresses high frequency components. This reduces of influence higher frequencies have in grouping harmonic signals. Such behaviour is predicted from psychophysical data. The model shows the sensitivity to phase relationships between the harmonics of different sounds. For the segregation of simultaneous vowels, one has to choose a lag of interest from the summation autocorrelation output. This necessitates the use of other mechanisms for testing each peak (the output of a two vowel sound contain more than two peaks) or the use of a-priori knowledge of signal content. From this perspective, the model seems useful as a feature extractor for global pitch cues which can be used by an architecture that can integrate the evidence from different features. By this it is meant that, contrary to the presumed intentions of the authors (i.e. using the model as a sole processor of pitch information), the model could supply one of several other cues regarding the possible existence of harmonic correlation in a region of sound. The point is that without the advantage of context there could be no way of telling how significant (or relevant) the spectral correlation at the model's output is. This point is directly related to the assumption of the researchers that some set of criteria exist for the separation of foreground from background sounds. Moreover, the incidental fact that the model attempts to separate a *two*-vowel compound into *two* states (fore- and background) is not realistic. Though it may be true that our attention mechanisms allow us to *listen* to at most one stream, we can still *hear* many more streams, and even sub-streams at the same time without 'throwing' all the non-attended events into the 'background'.

3.1.3 Wang D, Stream Segregation Based on Oscillatory Correlation[1]

- **Features used:** temporal and spectral proximity
- **Assumptions:** neural substrate evidence used literally for segmentation; streaming into at most two streams
- **Goals:** segmentation of symbolic tone input of sequential and simultaneous arrangements
- **Emergent Psychophysical properties**: apparent competition between spectral and temporal proximity

Wang (Rosenthal & Okuno 1998, chapter 6) proposes a neural-network based model for the primitive segmentation of auditory stimuli. The network is composed of an arrangement of single oscillators defined as feedback loops between excitatory and inhibitory units. Network rows represent time and columns represent frequency. Each oscillator in the network is laterally connected with its neighbouring oscillators. A global inhibitor receives excitation from each oscillator, and, in turn, inhibits each oscillator. Network input consists of input channels, each corresponding to a distinct frequency. Each oscillator within a frequency row in the network is directly connected by a delay line with the input channel. Delays are multiples of the basic delay duration (40 ms) and oscillator position along the row, so that each oscillator within a frequency row is activated at a specific point in time. In addition, each oscillator receives inputs from all other oscillators (see Figure 3-1). It is beyond the scope of the present work to get into the implementation details of the network; rather, the purpose of this review is to identify the motivation and usage of the model, as well as to try to assess how such a model would fit (or extend itself) into larger schemes.

[1] Computational Auditory Scene Analysis (Rosenthal & Okuno 1998), chapter. 6

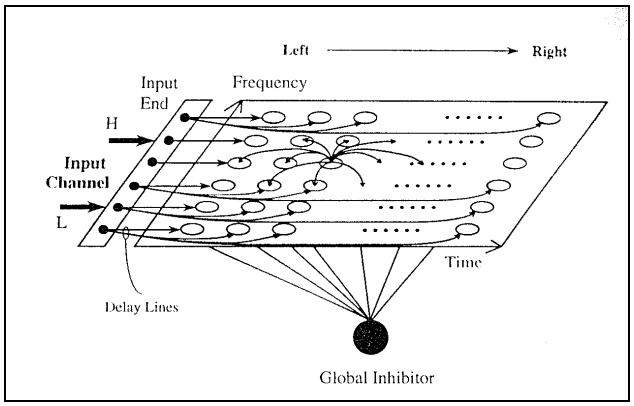

Figure 3-1: Diagram of the segmentation network. The connections from a typical oscillator in the network are shown in the figure, and those from other oscillators are omitted for clarity. As in the following figures, symbol H indicates an auditory input with a high frequency and L indicates an auditory input with a low frequency. [From Rosenthal & Okuno 1998, chapter. 6, p. 74]

For testing, a network of 15x30 oscillators was fed with symbolic representation sequences (i.e. not the result of features extracted by preprocessing from an audio fragment) of the psychoacoustic experiment depicted in Figure 3-2.

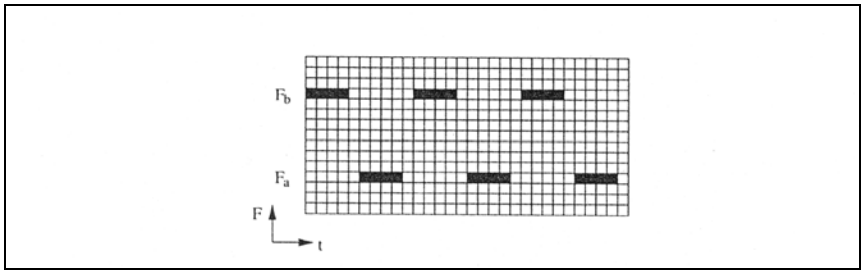

Figure 3-2: A stimulus pattern as mapped to the segmentation network at a specific time. This condition corresponds to fast presentation and large frequency separation. [From Rosenthal & Okuno 1998, chapter. 6, p. 76]

Based on the inter-synchronization between different frequency channels, the authors designed a segmentation criterion. The results of simulation of the first arrangement are shown in Figure 3-3. Oscillators within each frequency channel synchronised together, while not synchronizing with oscillators from different channels. Reduction of frequency separation resulted in the grouping of all tones together in agreement with psychoacoustic data. The authors describe a whole set of simulations of psychoacoustic experiments and show agreement with experimental data both for the grouping of sequential- and simultaneous-tones. Wang is motivated by the evidence for the existence of neural oscillators at thalamic and cortical levels of the auditory pathways. Moreover, he believes that the model can be extended for handling grouping based on other features than spectral and temporal proximity. It is not entirely clear, however, how such extensions are possible due to the need for interaction between the different features, possibly on multiple levels of abstraction. Further, is seems (at least from the available simulation results), that the model is limited to the production of at most two synchrony groups, so that each input tone must belong either to the one or to the other. It seems more likely, however, that the presence of more cues will require segmentation complexities far beyond those of two mutually exclusive streams.

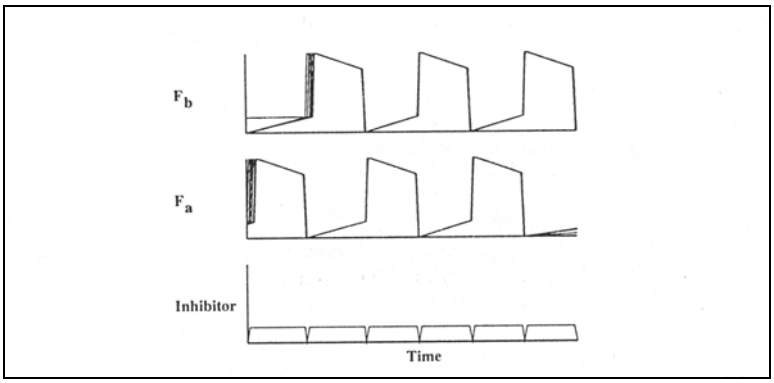

Figure 3-3: The combined activities of all the activated oscillators during one delay interval in the high- (F2) or the low-frequency (F1) channels, respectively. The ordinates indicate the normalized x values of the oscillators and the value of the global inhibitor. [From Rosenthal & Okuno 1998, chapter. 6, p. 79]

Wang's model belongs to the neural network paradigm. It is intriguing to notice how neural substrate evidence is used for devising neural network models. At the same time, the unbelievable complexity of an entire biological neural system is regarded only as an extension of simple networks. This is possibly the result of reasoning which posits that since the entire brain is composed of nothing but neural interconnections, it follows that complex neural systems are nothing but 'extensions' of simple ones. Such an approach proposes that perception processes never leave the substrate level, that is, that the brain processes sensory input on one level of abstraction. Alternatively, as I argue, one might accept the theoretical possibility that neural systems can rise above their substrate level and manipulate information at a more symbolic level. This does *not* necessary imply the symbolic level in the sense of traditional artificial intelligence models such as the General Problem Solver, see Russell & Norvig (1995) but rather in the same sense that a Turing machine can simulate neural network models and manipulate data at *their* level. In this view, physiological data changes its meaning from evidence for direct implementation to evidence of possible abstract information processing schemes. Finally, while the input of symbolic data is valid for the purposes of abstracting beyond feature extraction, and for building models for the investigation of grouping mechanisms, it is of importance to test the model's ability to cope with (at least some) irrelevant symbolic input, just as models which make use real sounds are tested also with so-called real-world fragments.

3.1.4 Cooke M., Modelling Auditory Processing and Organisation [1]

- **Preprocessing:** Gammatone filterbank, hair cell model
- **Features used:** instantaneous amplitude and frequency, FM, AM
- **Assumptions:** the existence of high level (see text) elementary units prior to scene analysis principles
- **Goals:** implementing a subset of primitive auditory scene analysis

Input monophonic audio signals are fed through a novel implementation of the Gammatone filterbank. The output of the filter represents the basilar membrane displacement with time at a particular place along the membrane. Cooke uses a static nonlinearity phase at the output of the Gammatone filterbank following physiological evidence for such nonlinear responses prior to neurotransmitter release. The final preprocessing stage is designed to reflect the hair cell to nerve fibre stage of auditory analysis. As Cooke attempts to stay as close as possible to physiological structures, he analyses these preprocessing phases and their implementations at depth. Consequently, Cooke summarises the properties of the hair cell model (see Cooke 1993, p. 29):

- *Spontaneous output rate with zero level stimulus*
- *Short term adaptation, with a time constant of 40ms*
- *Recovery to spontaneous level following adaptation with a time constant of 100ms*
- *Additive behaviour in response to stimulus increments*
- *Non additive behaviour in response to stimulus decrements*
- *Onset rate intensity function which exhibits sloping saturation at high intensities*
- *Adapted rate intensity function which saturates at a much lower level than that of the onset, and has a narrower dynamic range of 30-40 dB*
- *A ratio of onset to adapted rate which is an accelerated function of stimulus intensity with a value of 2.5 at a stimulus level of 60 dB*

The model's limitations, according to Cooke are due to the lack of fine temporal structure of the response. Cooke argues, however, that there is little evidence of the ability of the nervous system to preserve such information accurately. Therefore, the model is based on response envelope. Again, following physiological evidence, Cooke implements a model, which he believes, will support the implementation of some grouping principles. The model's parameters, however, are not extracted from psychoacoustic evidence, that is, the underlying assumption is, that at least for some degree, the model will behave according to psychoacoustic evidence *by* following physiological evidence. At the same time, Cooke admits that 'psychoacoustics provides

[1] (M. Cooke 1993)

(currently) the best guiding principles for making use of the representations provided by a model of the auditory periphery' (Cooke 1993, p. 30). It follows that Cooke regards physiologically accurate preprocessing phases as critical, but then allows his model to depart (partly for lack of concrete evidence of higher level processing) from the neural substrate.

Cooke proposes a model for the composition of the scene into 'time-frequency objects which characterise onsets, offsets and the movement of spectral components in time. The representation should allow further parameters such as amplitude modulation rate to be calculated'. Cooke's model proposes his Synchrony strands as the elementary auditory objects of auditory scene analysis. *'Each strand aims to define the time frequency behaviour of a single spectral component (e.g. harmonic or speech formant). Additionally, strands describe the local amplitude modulation rate, together with instant-by-instant amplitude fluctuations.'* The first stage is the estimation of the dominant frequency at each filter channel using median-filtered instantaneous frequency. Dominant frequencies for each channel are calculated each millisecond. The redundancy of neighbouring dominant-frequency channels is eliminated by combining similar estimates into so-called place-groups at each time frame. *'Finally, place groups are aggregated over time to produce an explicit time frequency description of auditory synchrony'.*

Thus, the model first attempts to build elementary units which could be used for the grouping principles as proposed by auditory scene analysis. These units provide information for grouping principles based on onset, offset, amplitude, frequency together with amplitude- and frequency-modulation. Place groups at joined together to form strands by following a strict algorithm. This, as Cooke puts it, is the way to achieve elementary units *'in the absence of any strong psychophysical data to guide an algorithm'.*

Place-groups at time t are aggregated into that strand at time t-1 whose trajectory passes sufficiently close.

This is achieved by computing the time derivatives of existing strands and predicting the expected place group continuing that strand in the next time frame. Using this estimation the following rules are evaluated:

1. *Place-groups which fall into any of the predicted regions are aggregated into the corresponding strand. It is possible that regions overlap, in which case one of the strands will be terminated.*

2. *Place-groups which do not fall into any such region form new strands; and*

3. *Strands which are unable to recruit place-groups are terminated.*

In the proposed model, only bottom up processing is implemented. It is not clear if different grouping principles could interact with each other to form higher-level groups. Cooke assumes the ability of the model to detect wrong strand composition 'as a result of grouping', that is, that when higher-level grouping principles fail to group a region they could 'rearrange' the primitive units. It is not yet clear, however, how such grouping principles can fail at all. According to Cooke, the ability of higher level processing to alter lower level ones does not fit into the model. In this context he quotes Marr's *principle of least commitment*, suggesting that it is preferable to avoid having to undo hard-and-fast decisions made by some earlier process. Cooke does not believe this to be a severe limitation, since *'objects which should belong together have several chances of being grouped as a result of the multi-level representational structure'.*

The model does not provide any means for the competition among elements. According to Cooke, there is too little psychoacoustic evidence to derive general rules for resolving competitions. He admits the possibility of using blackboard based systems for coping with the integration and competition needed for scene analysis. This side issue seems to me to be the key for tackling the problem of auditory as well as other domains of perception. See chapters 6 and 7 on *The Ear's Mind.*

The model evaluates different grouping principles by computing the different parameters for each strand. Such parameters are amplitude modulation and harmonicity. It is interesting to notice, however, that the search for parameters is taking place after the strands have already been formed. That is, strands are already formed without considering grouping parameters. It follows that grouping parameters are only used for the grouping of already formed units. This is best explained by Cooke (1993, p. 92):

'Synchrony strands arose as a result of grappling with the following problem: rules for auditory organisation are expressed in terms of frequency proximity, common AM rates, harmonicity, and so forth. But frequency of AM rate of what? ... Strands are an abstraction, created to support a functional description of auditory grouping. '

It is one of the goals of the present work to eliminate such hard-wired compound elementary units. They are compound since they contain a multitude of possibly contradicting pressures within, while at the same time they are treated as elementary since all the following processes use them as undividable building blocks. Moreover, the composition of such strands results in new problems such as the right way to make use of on- and off-sets. The search for the elementary units is covered in chapter 7.

3.2 General architectures for scene analysis

Framework models for the cooperative / competitive interaction of different grouping pressures

All sound separation systems based on perception assume a bottom-up or Marr-like view of the world. Sound is processed by a cochlear model, passed to an analysis system, grouped into objects, and then passed to higher-level processing systems. The information flow is strictly bottom up, with no information flowing down from higher-level expectations. Is this approach correct? [1]

Traditional top-down architectures will put bits of information together starting out with sensory input and working their way up by grouping elementary bits into chunks. In contrast, a top down approach will start out with one or more hypotheses for the existence of perceptual objects, and work down towards matching those objects with sensory input. The following models were designed to incorporate at least rudiment top down capabilities. Notice, however, that the ability to incorporate top down processing does not necessarily imply the use of schema based information. As I see it, the ability to incorporate pressures in all directions (up, down as well as 'sideways', that is, within the same level) is crucial for the implementation of working models of perception.

[1] Slaney M., in Computational Auditory Scene Analysis (Rosenthal & Okuno 1998), chapter. 3

3.2.1 Godsmark D. & Brown G. J., Context-Sensitive Selection of Competing auditory Organizations: A Blackboard Model [1]

- **Preprocessing:** Gammatone filterbank, hair cell model, spike generation and AVNC simulation
- **Features used:** Onsets, offsets, Frequency
- **Assumptions:** each feature expert produces multiple organisation hypotheses. These are later combined by other experts
- **Goals:** segmentation of pure tone input of into one or two streams. Implementing competition between the various feature experts.
- **Emergent Psychophysical properties**: gradual dominance of past winning hypotheses.

Godsmark & Brown (Rosenthal & Okuno 1998 chapter. 10) argue that resolving conflicting heuristics exclusively by bottom up means will prove futile. It seems that perception mechanisms resolve such conflicts by consulting the context. Such sensitivity to context requires the ability of different pressures to interact with each other at all levels of abstraction. It remains to determine the proper architecture which would support such functionality. In addition, architectures which allow for the autonomous workings of different pressures and their interaction will lend themselves more easily to future expansion. Following a Gammatone filterbank phase, the input signal is fed into the Meddis hair cell model producing the probability of a nerve spike occurring at each instant in time. Further preprocessing is applied which mimics some activities of the auditory nerve and the anteroventral cochlear nucleus. These approximate some convergence and inter-spike correlation issues of a per channel population of 20 nerve fibres. Two algorithms are employed for the computation of the required features. The first makes use of a summation across filter channels of the inter spike delays, thus following the phase-following tendencies witnessed in the auditory nerve. This arrangement enables the estimation of frequency and periodic related features. The second algorithm computes on- and off-sets, and involves the computation of average firing rates across fibres for each filter channel. Although preprocessing choice is motivated by physiological evidence, it is not quite clear how the estimation of frequency, onsets and offsets gains from such an approach. Is it possible that the brain itself takes a detour for evolutionary or anatomical reasons on the way to compute what we could do in a straightforward way?

Following feature estimation, the model builds a representation based on Cooke's synchrony strands (see section 3.1.4). The authors name these structures components and regard them as 'computational convenience'. Components are built by evaluating the following rules for each filter channel:

1. *If an onset is detected, start a new component and flag it as possessing a definite onset.*
2. *If an offset is detected, terminate the component and flag it as possessing a definite offset.*
3. *If a significant (determined by a threshold) change in frequency (periodic response) is detected, start a new component and label it with the new frequency.*

[1] Computational Auditory Scene Analysis (Rosenthal & Okuno 1998), chapter. 10

4. *If a region has a firing rate of zero, mark it as silence and ignore it in all future processing.*

Further, when a new component is started in one channel, all other channels get a new start at that point in time to allow possible inter channel capture to occur. It is obvious from the above rules that no distinction is made between the estimation of an onset feature (that is, a region with a high response rate as evidenced from direct sensory data) and the psychophysical concept of an onset (that is, a starting point of an object in the real world as evidenced by context). On the necessary distinction between the two, see chapters 6 and 7.

The authors are motivating their grouping strategy with the following observations:

1. *Organisations can be formed retroactively; that is, an organisation does not immediately have to be imposed upon components.*

2. *Organisation is context sensitive; that is, it is not just the relationship between adjacent components that determines their organisation – their relationship to surrounding components must also be considered.*

The strategy chosen by Godsmark & Brown is to retain all potential organisations 'in the hope that the evolving context will provide enough additional evidence to resolve the conflict' (Rosenthal & Okuno 1998, chapter. 10, p.145). This is of course one of many possible ways to deal with context sensitivity. Notice also, that the two points above are not mutually exclusive, and that if delayed decision is allowed, then retroactive behaviour could be attributed to context sensitive behaviour. To allow for delayed decision, the model employs a sliding temporal window, the organisation hypothesis region, within which alternative organisations are allowed. The ultimate organisation is imposed on components once they leave that region. For the sliding window, a temporal duration of 250 ms was chosen based on the retroactive working of perceptual restoration. The authors have implemented a blackboard architecture which scores and imposes different organisations for components within the temporal window. Such organisations form the basis of the next cycle of grouping. The model consists of four distinct phases.

- *Organisation hypothesis formation.* Formation experts decide whether hypotheses made in the preceding window should be continued. If not, new hypotheses will be created. For each grouping principle, there is one expert which applies its heuristics autonomously. At this phase then, we have a set of hypotheses for each expert. It is interesting to note that single experts, such as the frequency proximity expert used as an example in their paper (Rosenthal & Okuno 1998, chapter. 10, p. 147) create multiple hypotheses even without the inclusion of other experts. On the other hand, hypotheses to which any single expert objects, are guaranteed not to be imposed as a final organisation. This approach, plausible as it may be, is not in line with assumptions made in psychoacoustic experiments aiming at isolating certain heuristics. The tendency of single pressures to produce single interpretations should follow from the acceptance of the success of such experiments at isolating the working of single heuristics. Godsmark & Brown, point out that 'it is difficult to conceive of an experimental paradigm which would allow the contribution of a single grouping principle to be isolated' (p. 150)

- *Organisation coalition.* A single expert, the organisation coalition expert, merges the sets of organisation hypotheses to form a unique set of hypotheses by removing redundant ones.

- *Organisation evaluation.* At this phase, each hypothesis is evaluated by a range of evaluation experts, one for each grouping principle. The scores from the various experts are then combined to produce the so-called plausibility index which is taken as an additional seed when evaluating the next window. Notice that for the correct working of this scheme, the

windows have to progress in steps no bigger than single components, and they have to overlap each other in ranges which permit useful evaluation.

- *Organisation selection*. The organisation corresponding with the highest plausibility index will be imposed on components exiting the hypothesis window.

Though I fully support the need for context sensitivity, I wonder whether the implemented scheme actually follows context, or just maintains continuation of interpretation temporally. It is not clear, for instance, whether or not more than two simultaneous streams could be formed. If one frequency proximity expert works on the entire window, how does this expert handle the context of the components in its region? In this sense, it seems more likely that context differences arising from different experts might be taken into account, but inter-feature context is handled globally. Furthermore, the need for absolute proximity metrics arises as the relevant experts score components to each other without comparing them with distances to other components. This is one example of the distinction between global evaluation and context evaluation. That is, the definition of proximity should not rely on absolute measures, but on relative proximity of other components in the region. In this way, proximity scores between two components will depend on the position of other components in the same area. See chapter 7. For each grouping principle, two evaluation metric curves are used. One curve is used for scoring a hypothesis which groups two components in the same stream, while the other is used when the two are segregated. Preliminary results show agreement with psychoacoustic results of the given experiment. It is interesting to note, that the use of the winning hypothesis as a seed for successive windows results in the gradual dominance of that hypothesis in successive windows.

3.2.2 Klassner F. et al, The IPUS Blackboard Architecture as a Framework for Computational Auditory Scene Analysis [1]

- **Preprocessing:** time domain envelope, spectrogram

- **Features used:** spectral features: peaks, contours, spectral band activity, time-domain features: energy shifts, up (on) and down (off), average energy etc.

- **Assumptions:** fixed signal processing front end is not flexible enough for perception of real world scenes.

- **Goals:** implementing a general purpose blackboard system for analysing complex signal scenarios

The IPUS (Integrated Processing and Understanding of Signals) architecture was designed for coping with complex signal environments with unpredictable sources and 'simultaneous occurrence of objects whose signal signatures can distort each other'. The IPUS model in this review represents a scale up of the original architecture and supports a library of 40 sound models. Currently it is capable to analyse scenes containing up to four sounds. Ambiguous data at the output of the signal processing front-end could be reprocessed by pressures from higher-level

[1] Computational Auditory Scene Analysis (Rosenthal & Okuno 1998), chapter. 8

processes, including the alteration of processing parameters to dynamically adjust for context. Additionally, distinct regions of input data can be explored due to expectations arising from higher level interpretations.

For each block of input data, the model iterates a loop. The starting point is the signal-processing phase with default parameters. Next, discrepancies among the output of different signal processing levels, the model's constraints, and expectations are detected. If such discrepancies are detected, a 'diagnosis process attempts to explain them by mapping them to a sequence of qualitative distortion hypotheses'. The loop ends with a signal reprocessing stage that proposes and executes a search plan to find a new front-end setup which will result in reduction in discrepancies. When competing top-level interpretations with equal scores are encountered, a special process is called to select and execute a reprocessing plan to discriminate among the alternatives. This process is not restricted to the current data block. Reprocessing and evaluation can be applied to past blocks as well as postponed for later, when specified conditions are met. For applications based on this model, see next section.

3.2.3 Ellis, D. P. W., Prediction-Driven Computational Auditory Scene Analysis for Dense Sound Mixtures [1]

- **Preprocessing:** Cochlea filterbank, rectified and smoothed time-frequency intensity envelope, short time autocorrelation, periodogram
- **Features used:** Onsets, periodicity
- **Goals:** segmentation of real world scenes into a predefined set of sound elements. Implementing a prediction-driven system with an abstraction of auditory events as an internal world model.

Ellis (1996) posits that a working scene analysis engine would require the ability to handle any kind of sound and any competition among conflicting hypotheses. In addition, and in contrast the data-driven architectures, he argues, that only a prediction-driven scheme would be able to tackle 'context dependent inferences in the face of interference'. Ellis equates prediction driven activity with top down processing. In my opinion, this is misleading as one could implement non prediction-driven mechanisms using top-down techniques as well. In such implementations, higher-level structures can break down their lower-level constituent structures and push towards reorganisation at lower levels. This working, however, implies neither the use of prediction nor knowledge of a specific auditory domain. Ellis lists the working principles of his model as follows:

- *Analysis by prediction and reconciliation*: The central operating principle is that the analysis proceeds by making a prediction of the observed cues expected in the next time slice based on the current state. This is then compared to the actual information arriving from the front-end; these two are reconciled by modifying the internal state, and the process continues.

[1] (Ellis 1996)

- *Internal world-model:* Specifically, the 'internal state' that gives rise to the predictions is a world model – an abstract representation of the sound-producing events in the external world in terms of attributes of importance to the perceiver. Prediction must work downwards through the abstraction hierarchy, resulting eventually in a representation comparable to the actual features generated by the front-end.

- *Complete explanation:* Rather than distinguishing between a 'target' component in a sound mixture and the unwanted remainder, the system must make a prediction for the entire sound scene – regardless of which pieces will be attended to subsequently – in order to create a prediction that will reconcile with the observations.

- *Sound elements & abstractions:* The bottom level of the internal world model is in terms of a mid-level representation of generic sound elements. In view of the requirement for complete explanation, these elements must encompass any sound that the system could encounter, while at the same time imposing the source-oriented structure that is the target of the analysis. Higher-levels of the world model are defined in terms of these elements to correspond to specific (constrained) patterns known to the system.

- *Competing hypotheses:* At a given point in the unfolding analysis of an actual sound there will be ambiguity concerning its precise identity or the best way to interpret with the available abstractions. Since the analysis is *incremental* (maintaining at all times a best-guess analysis rather than deferring a decision until a large chunk of sound has been observed), alternative explanations must be developed in parallel, until the correct choice becomes obvious.

The main points relevant to my proposed model are:

- The implemented prediction mechanism works by comparing lower level input cues to lower level cue predictions. How does this strategy compare with prediction evaluation at the abstract level? In other words, the model predicts the existence of specific cues at the input rather than pushing towards a preferred (context dependent) interpretation of such cues.

- Ellis argues for the need to predict all the constituent components in the scene, rather than only separating a target component from its interfering background. Any other arrangement would break down when the auditory scene is more complex, as it always is, and would need an a priori assignment specifying the scene's composition.

- The model maintains multiple hypotheses at any time which direct its future hypotheses. An alternative approach is discussed in (chapter on copycat).

  Ellis defines three basic sound structures: *noise clouds* are those patches of aperiodic regions having slowly varying envelopes. These are needed as the model attempts to explain every region of the scene. *Transient clicks* are sounds with abrupt onsets of aperiodic structures which give rise to a different perception than the more continuous noise clouds. Finally, *Wefts* form the model's representation of wide and periodic energy. The model's blackboard architecture is based on the IPUS model (see section IPUS). The model works by comparing current input features with the predicted features from past evaluations. This is done by following the parameters of the sound elements used in past hypotheses. By comparing the predicted behaviour of these parameters with the actual ones, the model can take such actions as continuing the current hypothesis with the possible alteration of some parameters, terminating it, or generating a new set of alternatives.

The presented results show that the model is able to segment a real world sound scene into the predefined set of sound elements. The segmentation agrees with human tendencies to notice the predefined elements. By using the blackboard architecture, it was possible to define sound elements with more abstract nature and overlapping characteristics.

4 Analogy Making As Perception

Copycat

High-level perception is not the result of using a set of serially applies, conscious mental rules, but rather that it emerges as a statistical outcome of large numbers of independent activities occurring in parallel, competing with and supporting one another, and influencing one another by creating and destroying temporary perceptual constructs. Such a system has no global executive deciding which processes to run next and what each should do. Rather, all processing is done locally by many simple, independent agents that make their decisions probabilistically. The system is self-organizing, and coherent and focused behaviour is a statistically emergent property of the system as a whole… (Mitchell 1993)

Starting with an introduction to the domain of auditory perception, I have presented the various sources of data available for laying out the fundamental requirements for modelling perception. These include data from psychoacoustics, physics, and anatomy. I have included a short critical survey of models in the field of computational auditory scene analysis and discussed those implementation features, which enable models to reach some auditory functionality. Some of the shortcomings I have described made clear that an architecture is needed, which supports emergent processes, cooperation and competition in several layers of abstraction, extendibility, and context sensitivity. In this chapter, I describe the Copycat model (see Mitchell 1993) and argue that, when abstracted from its original micro-domain and specific sort of analogy-making paradigm, it promises to support the implementation of emergent, context sensitive processes in general, and (human) perception mechanisms in particular. Later in this work, (see chapter 6) I describe what makes Copycat a suitable platform for modelling perception and the steps needed for transforming it from its original form into such a perception engine. But first, what is Copycat?

4.1 Copycat: a computer model of analogy making

'Copycat's task is to use the concepts it possesses to build perceptual structures - descriptions of objects, bonds between objects in the same string, groups of objects in a string, and correspondences between objects in different strings - on top of the three "raw", unprocessed strings given to it in each problem.' (Mitchell 1993)

The idea behind Copycat is to mimic the emergent behaviour of natural complex systems, believing that such processes play an important role in human perception mechanisms. For a history of Copycat and earlier models of perception by Hofstadter et al., see (Hofstadter 1995). Hofstadter's intuition into the working of human cognition in general, and the processes of perception, and analogy-making in particular, has led him to believe that the puzzling flexibility of the human mind even at the subconscious levels of perception (i.e. closer to sensory input) could not be emulated without the use of emergent properties. Such flexibility of mind and the existence of what he calls *fluid concepts* could not be programmed by fully deterministic processes that work at the same level of abstraction of the outcome itself. Instead, microscopic, nondeterministic, local processes should interact with each other with no central control. The macroscopic outcome of such activity is *emergent* and viewed as the consequence of such microscopic interactions. One of the first metaphors used in this context was *Aunt Hillary* (Hofstadter 1979) contrasting the microscopic interaction of individual ants within an ant colony with the vast repertoire of sophisticated macroscopic behaviours that the colony displays as a whole. The difference between the simplicity of ant behaviour at the individual level, and the complexity of the resultant emergent behaviour of an ant colony as a whole have fascinated researches in the fields of biology and complex systems throughout the years. How could organised behaviours such as resource management (bees and ants), the construction and

maintenance of complex structures (termites) and schooling behaviour (fish and birds) be achieved with no centralized control? Moreover, such behaviours are flexible and react to different circumstances 'fluidly'. That is, solutions emerge which were not anticipated, or programmed in advance. The term fluid brings about the imagery of water taking the form of the container it is in. Other macroscopic characteristics of fluids such as their boiling point and viscosity come about as emergent properties resulting from the local random interaction (under the basic laws of physics) of their constituent molecules. No fluid has a repertoire of macro shapes pre-programmed; molecules are not even 'aware' of others except for their nearest neighbours. Indeed, searches have been conducted for years for the illusive central communication of fish schools, bird flocks and termite nests. The belief was that with no central directions, such complex order could not be reached, and sustained. On the fascinating world of the emergent behaviour of social insects, see (Marais 1937, Camazine et al 2001). Microscopic behaviour giving rise to ordered macroscopic phenomena can even be seen in chemical systems. Examples of such systems can be found in (Prigogine 1984).

As Hofstadter (1995, p. 208) puts it, '*a casual glance at the project might give the impression that since it [Copycat] was specifically designed to handle analogies in a particular tiny domain, its mechanisms are not general. However, this would be a serious misconception. All the features of the Copycat architecture were in fact designed with an eye to great generality… the Copycat project is not about simulation analogy making per se, but about simulating the very crux of human cognition: fluid concepts. The reason the project focuses upon analogy-making is that analogy-making is perhaps the quintessential mental activity where fluidity of concepts is called for…*' Hofstadter goes on saying that (p. 210) '*…one of the main ideas of the project is that even the most abstract and sophisticated mental acts deeply resemble perception…The essence of perception is the awakening from dormancy of a relatively small number of prior concepts – precisely the relevant ones. The essence of understanding a situation is very similar; it is the awakening from dormancy of a relatively small number of prior concepts – again, precisely the relevant ones – and applying them judiciously so as the identity the key entities, roles, and relationships in the situation*'. Interestingly, Copycat's architecture was inspired by an early model of speech recognition, the Hearsay project, which used a black board like system. Though it might seem that speech perception has little to do with the Copycat domain, I argue later on (chapters 6 and 7) that low-level perception and higher-level analogy making share much of their mechanisms and, consequently, their fluidity.

In Copycat, analogies are made between strings of letters. Their respective structures mould the emerging concepts which are built on them, so that 'fitting' concepts take shape. These concepts fit the context in such a way, that the resulting analogies between the given letter-strings make sense. As I have described (see chapter 2), I view the primitive cues of perception (i.e. sensory data) as the base from which context is continually influencing the emergence of concepts. The fact that context is made up from various contradicting (possibly sensory and multi-modal) pieces of data, means that the interpretative result bubbling up is an interpretation of reality, neither *right* nor *wrong*, but one which does *more or less* fit the range of internal active concepts and external input.

Copycat's micro domain functions only as a testing environment for a more general analogy-making mechanism. But the ability to reach such fluid behaviour relies on much more internal activity than can be seen from the surface. As Hofstadter puts it, '*One of the central goals of the Copycat architecture is to allow many pressures to simultaneously coexist, competing and cooperating with one another to derive the system in certain directions. The way this is done is by converting pressures into flocks of very small agents…, each having some small probability of getting run…over time the various pressures thus 'push' the overall pattern of exploration different amounts, depending on the urgencies assigned to their Codelets.*' It is this activity - the constant workings of different micro-flocks in parallel, pushed around by pressures, redirecting in unpredictable ways, allowing any possible interpretation to gain in strength, yet tending to follow a dominant (i.e. fitting) one - which makes Copycat's behaviour seem so natural. Indeed, it is when watching Copycat while it goes about looking for an answer to a problem that one realises how natural its behaviour is. Copycat is an attempt to combine

emergence with an abstract notion of what cognitive concepts are, and to reach a "cognitive community" which comes up with answers to its analogy-making problems. The current proposed model, *The Ear's Mind*, is intended as a new domain (auditory perception) for testing the ability of architectures similar to Copycat to emulate human perception, but by that, I certainly do not imply that such mechanisms will suit only auditory perception. The very reason for that is my belief that the crux of cognitive activity lies in its emergent behaviour. In this sense, the auditory domain is no different from any other.

It is interesting to note Hofstadter's views regarding Copycat's emergent behaviour.

'There is one further aspect of shadedness in Copycat that is not localized in a single component of the architecture, and is somewhat subtler. This has to do with the fact that, over time, higher-level structures emerge, each of which brings in new and unanticipated concepts, and also opens up new and unanticipated avenues of approach. In other words, as a run proceeds, the field of vision broadens out to incorporate new possibilities, and this phenomenon feeds on itself: each new object or structure is subject to the same perceptual processes and chunking mechanisms that give rise to it. Thus there is a spiral of rising complexity, which brings new items of ever-greater abstraction into the picture "from nowhere", in a sense. (Hofstadter 1995, p. 267) Hofstadter follows with a list of consequences of this behaviour:

Dynamic emergence of unpredictable objects and pathways

- *Creation of unanticipated higher-level perceptual objects and structures*
- *Emergence of a priori unpredictable potential pathways of exploration (via creation of novel structures at increasing levels of abstraction)*
- *Creation of large-scale viewpoints*
- *Competition between rival high-level structures*

This behaviour is, in my opinion, exactly what a computational model of perception needs. But before I propose a way to reach such goals (which I describe in chapters 6 and 7), I sketch the internal workings of Copycat. This will prove handy later on when I describe the alterations needed for stripping down Copycat's specific domain and getting it ready for primitive auditory scene analysis. But first, very briefly, what is Copycat's original micro-domain?

4.2 The Copycat domain

The Copycat programme is all about the implementation of 'statistically emergent active symbols' (Hofstadter 1995, p. 205). All it sets out to achieve is a working architecture for testing whether nondeterministic activity of micro-agents could bring about order at a higher-level and whether such order could fit a given context with no *a priori* knowledge. But to do that, one needs a testing ground for verifying results, and even more importantly, for investigating how the programme's answers emerge. Hofstadter chose for the Copycat domain, a simple domain with a limited number of elements. However, in a cognitive sense, the domain is quite "real" since various alternative "correct" answers are always possible. Some give humans more "satisfaction", while others seem shallow. In other words, different people may find different solutions in this game. People sense the creative process of finding insightful analogies: ones that bring about deeper contextual relations in a given problem.

The game goes like this: say the letter-string ***abc*** was changed to ***abd***, how would you change the letter-string ***ijk*** 'in the same way'? The fact that you are not given the same letter-string obviously renders the phrase 'in the same way' into 'in an analogous way to what I did'. Now, most people would argue, that since I have changed the rightmost letter ***c*** in ***abc*** to a ***d***, the alphabetic successor of ***c***, that the idea of "doing the same" would be to change the rightmost letter in ***ijk*** into *its* successor, resulting in ***ijl***. Although other, less compelling answers could be found, such as ***ijd*** (change the rightmost letter into a ***d***), it is because the letter-strings ***abc*** and ***ijk*** share so much *structure*, that only a few "good" answers exist. As a simple example of the interesting possibilities of the domain, Hofstadter alters the above problem into the following:

Suppose the letter-string ***aabc*** were changed to ***aabd*** ; how would you change the letter-string ***ijkk*** in "the same way"?

Now things can get a little more complex. The answer depends on how you *interpret* the problem. The simple solution from the above problem can still be applied here: change the rightmost letter into its successor. That would result in the letter-string ***ijkl***. The new context, however, gives rise to pressures which did not exist in the first problem. As Hofstadter remarks, applying the above solution in this case does seem a bit crude to most people, because it ignores the obvious fact that the ***k*** in ***ijkk*** is doubled. There is a pressure to "take the two ***k***'s together", that is, not to break up the two ***k***'s unit. So many of us will feel more comfortable to *bend* the rigid rule *replace the rightmost* ***letter*** *with its alphabetic successor* into *replace the rightmost* ***letter-group*** *with its alphabetic successor*. This gives the answer ***ijll***. It seems that the concept *letter* has *slipped* under pressure into the related concept *group of letters*. Seen from this perspective, the more complex analogy leading to ***ijll*** (given ***aabc*** → aabd) is not *different* in any significant way from the analogy made in the first problem (***abc*** → ***abd***). Both analogies make use of the available pressures in the context of the given problem. The difference in the answers is made possible by the fluidity of the concept *letter* and the ability of pressures to influence the direction taken by the programme. Moreover, Copycat is not programmed with a list of rules such as "replace the rightmost letter with its alphabetic successor", or "replace the rightmost letter-group with its alphabetic successor". If that were the case then Copycat would be reduced to nothing more than a brute-force implementation. It would then search through a set of rules and eventually choose an answer from a list of possible "good" answers. Of course, the fact the Copycat has no predefined list of answers does not mean that it can come up with answers that lie beyond its capacity. It is limited by many factors including even the type of answers it can give (i.e. the template *replace* ___ *with* ____). Take the following problem: ***abc → abd, kji → ?*** Copycat is able to answer the problem with ***kjh*** only because it has (fluid) concepts such as *successor* and *predecessor* but nowhere is it programmed with heuristics which for instance check for whole string reversals. Rather, it is the result of the simultaneous pressures, (if they happen to be discovered) of noticing that letters in corresponding positions in different strings have *opposite* relations within their corresponding strings which can bring about the activation of the concept *opposite* in the answer. ***kji*** is then seen as a group of alphabetic predecessors related to the alphabetic successor group ***abc***. Doing "the same thing" to the replacement rule brings about the answer "replace the rightmost letter with its predecessor". This seems deeper than other answers because it makes use of all the existing pressures in this problem. Needless to say, Copycat has got to have a basic set of primitive concepts and an idea of how they inter-relate (e.g., that *successor* is the *opposite* of *predecessor*). It also needs to understand its domain (i.e. know that it has to come up with a letter string which relates to the third string in "the same way" in which the second string relates to the first string). Copycat's agents have to be able to discover simple relations between letters, or to build groups of letters etc. These capabilities are, however very primitive in comparison to the integrated emergent result. This is analogous to a community of termites, where each individual has only a basic knowledge of its role and where the chemical means of communications are hardwired. The emergent behavioural fluidity of the community as a whole, however, is not pre-programmed, and is therefore non-deterministic. Copycat's non-determinism means that at every moment in time, it might discover some new evidence and alter its future path, leading to new ways of

interpreting the problem. When the pieces fit together well, the tendency of staying on the same path is strong. Weak structures tend to bring about more randomness into play which opens up new ways of investigation. Consequently, Copycat may give different answers each time it runs on the same problem. The above problem, for instance - ***abc → abd, kji → ?*** **,** when made to run a 1000 times (Hofstadter 1995, p. 239 and Mitchell 1993), came up with the answer ***kjh*** 547 times. It answered 232 times with ***kjj***, and 204 with ***lji***. ***kjd*** was the outcome 15 times, while both ***dji*** and ***kji*** occurred just once. Moreover, the answer ***kjh*** was not only the most common answer, but also had the lowest *temperature,* signifying a deeper, better answer (on the purpose and function of temperature, see Mitchell (1993)). The answers ***kjj*** (232 times), and ***lji*** **(**204 times), for instance, occurred nearly the same number of times, but ***lji*** had a much lower average temperature.

Would it be possible to implement the Copycat domain by deterministic, brute-force means, without resorting to the complex architecture? Sure, but that would be missing the point altogether. The fact is that the Copycat domain (even if it does have a fascinating range of possible analogy problems and unexpected "beautiful" answers) is isolated and small, and it is therefore theoretically entirely possible to implement a traditional programme based on search algorithms which would do quite a good job. But that would be programming for the domain rather than *using* the domain as testing grounds for the real goal – coming up with an architecture which supports the emergence of fitting structures using a community of non-deterministic agents and fluid concepts. The idea is that only such an implementation would be capable of coping with more "real – world" domains. In the next section, I describe briefly the internal design of Copycat. A thorough discussion, together with comparisons with other related models can be found in (Mitchell 1993).

4.3 Copycat's Architecture

Copycat relies on the interaction of various mechanisms, which Mitchell (1993) summarises as follows:

1. *Concepts consisting of a central region surrounded by a halo of potential associations and slippages, in which the relevance of the concept and the proximity to other concepts change as the process of perception and analogy-making proceeds*

2. *Mechanisms for probabilistically bringing in concepts related to the current situation and conceptual slippages appropriate for creating an analogy*

3. *A mechanism by which concepts' relevances decay over time unless they are reinforced*

4. *Agents that continually look for interesting properties and relationships among objects and structures in a working area and attempt, on the basis of their findings, to build new descriptions, bonds, groups, and correspondences*

5. *Mechanisms for applying top-down pressures from concepts already deemed to be relevant*

6. *Mechanisms allowing competition among pressures*

7. *The parallel terraced scan, allowing rival views to develop at different speeds*

8. *Temperature, which measures the amount of perceptual organization in the system (low temperature meaning a high degree of organization) and which, on the basis of this value, controls the degree of randomness used in making decisions.*

The main elements of Copycat's architecture consist of the *Slipnet*, *workspace*, *Coderack* and *temperature*. These structures contain Copycat's input, output, state and concepts. The *Slipnet* holds Copycat's concepts in the form of a network of nodes and links. Each concept, however, is defined dynamically as a region around its central node rather than as a single node. Potentially, any concept includes all the nodes directly attached to its central node. As *Codelets* (simple agents, each dedicated for the detection of simple features or relationships, such as when two letters or small numbers are successors of each other) find matching instances in the input, they activate their corresponding concept-node in the *Slipnet*. The activation of each node affects the probability of Codelets to consider it while working on a solution to a given problem. That means, that throughout the run of the programme, the dynamic activation levels of nodes will allow different concepts to be present, influencing the course of processing. When a node (such as *successor*) is fully active and under the right circumstances, it can decide to launch more agents to search for more instances of itself. *'All concepts have the potential to be brought in and used. Which ones become relevant, and to what degree, depends on the situation the program is facing'*. At the beginning of a run, a predefined set of *Codelets* are put on the *Coderack*, the structure from which all *Codelets* waiting to be launched on the *workspace*, are "hanging". Deciding which *Codelet* to launch at any given moment is a stochastic function which relies on various parameters, including the current '*temperature'* and the urgency with which each *Codelet* was put on the *Coderack*.

When a node in the *Slipnet* is activated, it spreads its activation to neighbouring nodes. This spread is a function of the proximity (i.e. link lengths) between neighbouring nodes. As a result, an active node will spread some activation to other relevant (i.e. close) concepts. The activation levels of all nodes are updated at predefined intervals (measured in the number of *Codelets* launched between successive updates). Activation levels decay automatically with time reactivation occurs. This mechanism ensures that concepts which were relevant earlier in the run, but no longer seem relevant now, loose some probability to be launched in the near future. Low-probability *Codelets* of all types, however, are constantly added to the *Coderack* so that the chance of rediscovering a feature that may turn out to be relevant is always there.

The rate of activation decay is a function of the node's *conceptual depth*. This predefined measure signifies the generality, or abstractness of a concept. The idea behind it is that the more abstract a concept, the slower it should be disposed of. For example, the concept *opposite* in Copycat's micro-domain is deemed deeper than that of *successor*. Consequently, once active, *opposite* will lose its activation at a slower rate. In Mitchell's words, '*the depth of a concept is how far that concept is from being directly perceptible in situations*'. In the domain of auditory scene analysis, it might be said, that the concept of speech formats (if defined as a node in the Slipnet) is deeper than that of a strong temporal derivative in the input signal. The reason is that speech formants are deeper conceptually, and consist of more than a single feature in the input signal. Subsequently, the relevance of the derivative may die out quicker than that of the formant if no further evidence in the input is found. This mechanism drives the model to use deeper concepts for explaining the structure of a given situation once such concepts are found.

The *conceptual proximity* of two nodes in the Slipnet depends on the length of the link between them. Although links have pre-assigned lengths, some are labelled, by other nodes. When such label nodes become active, they shrink all the links they label. For instance, because in Copycat, the link between *rightmost* and *leftmost* is labelled by the *opposite* node, when the concept *opposite* is active, it renders *rightmost* conceptually closer to *leftmost* as well as all the other nodes that are linked by the *opposite* node. This makes two such nodes prone to *slippage*, that is, that the

perception of one, will lead by association to the other. See the above example ***abc → abd, kji → ?***

The *workspace* is where perceptual structures are built on top of the input. The model makes use of six types of structures:

1. Descriptions of objects such as *first* for the letter 'A'
2. Bonds representing the relations between objects in the same string
3. Groups of objects in the same string
4. Correspondences between objects in different strings
5. A *rule* describing the change from the initial string to the modified string
6. A *translated rule* describing how the target string should be modified to produce an answer string

All these structures are dynamically built (or broken) on the workspace by *Codelets*. *Bottom-up Codelets* look for any structure they might find. Such a *bottom*-up *Codelet* first "land" on a letter (if it is a *description Codelet*) or two letters (e.g. a *bond Codelet*) and looks if it can propose a structure to be built on (or between) the letter(s) it landed on. *Top-down Codelets* look for instances of particular active nodes. That is, such *top-down Codelets* are sent by active concepts to seek other instances of themselves. Since, as describe above, the activation levels of nodes decay when *Codelets* fail to reactivate them, it follows that when such nodes are no longer active, they cannot launch *top-down Codelets*. The opportunity remains, however, for *bottom-up Codelets* to discover new instances of such nodes and contribute to their reactivation. In other words, the programme spends most of its time on concepts which were dynamically deemed relevant, while still allowing new concepts to be perceived. At the same time, concepts which are no longer relevant die out and no longer launch *top-down Codelets*. This allows the programme to use its dynamic resources on activities that are more relevant.

The tasks of searching for, and building features is broken into several steps to allow for more promising (i.e. relevant), or urgent structures to be carried out faster then others. A series of *Codelets* run in turn for each structure. Each *Codelet* in the series decides probabilistically, based on progressively deeper estimations of the structure's promise, whether or not to call a follow-up *Codelet*. If it decides to go on and call the next *Codelet* in line, it gives its successor an *urgency,* which will probabilistically determine its chances to be launched, and puts it on the *Coderack*. At every step, the *Coderack* launches one chosen *Codelet*. Choosing a *Codelet* is done probabilistically biased by the relative urgencies of all *Codelets* on the *Coderack*. This arrangement differs from that of systems such as Hearsay II where, at each step, the action with the highest priority is launched. The probabilistic behaviour, together with the action split into a series of three *Codelets* (which find, test, and build structures), results in a dynamic parallel configuration, in which more urgent pressures are dealt with faster than less urgent ones, though the lower-urgency pressures always have a chance to run. The urgencies of *bottom-up Codelets* are fixed, while the urgencies of *follow-up Codelets* posted by other *Codelets* are the result of the evaluation done by the calling *Codelet*. A *top-down Codelet* (posted by an active node in the Slipnet) gets an urgency which is a function of the activation level of the posting node. Thus, as Mitchell puts it, '*the Codelet population on the Coderack changes, as the run proceeds, in response to the system's needs as judged by previously run Codelets and by activation patterns in the Slipnet which themselves depend on what structures have been built. The speed of a structure-building process emerges dynamically from the urgencies of its component Codelets. Since those urgencies are determined by ongoing estimates of the promise of the structure being built, the result is that structures of greater*

promise will tend to be built more quickly than less promising ones'. (Mitchell 1993, p. 43). Although the existence of large-scale processes is vital to the correct working of the model, such processes are not pre-planned. Large-scale activity can only be determined *a-posteriori*, that is, an observer could only conclude that such a large-scale process has taken place *after a run* but there is no direct programming of large-scale pressures in Copycat. Such processes are emergent rather than planned, and dynamically reflect the context discovered by the model.

Since Copycat seeks structure rather than specific organisational constructs, it follows, that a mechanism is needed for the evaluation of structure quality. Such evaluation is needed if the programme has to produce an answer at the end of its run. This is the purpose of the *temperature* mechanism. By measuring the degree of *perceptual organisation* in the system as a function of the amount and quality of structure built so far, it can reach a low enough '*temperature'* (indicating strong structures) and decide to shut down the structure-building processes and switch to answer construction. In that sense, the final *temperature* can be interpreted as the quality of the answer, as deeper structures result in answers which involve more abstract perceptions than superficial ones. In addition, the *temperature* measure controls the degree of randomness used in making decisions throughout the run. This behaviour is motivated by the notion that the stronger structures are, and the deeper the structure is, the closer the programme is to a good solution. On the other hand, if no deep structures have been found, the programme's randomness grows, pushing the search in new directions and giving chance to novel solutions. Mitchell relates the *temperature* mechanism to the following apparent paradox: '*you can't explore every possibility, but you don't know which possibilities are worth exploring without first exploring them. You need to carry out some exploration in order to assess the promise of various possibilities, and even to get a clearer sense of what the possibilities are. You must be open-minded, but the territory is so vast that you cannot explore all of it. In Copycat, the fact that Codelets are chosen probabilistically rather than deterministically allows the exploration process to be a fair one, neither deterministically excluding any possibilities a priori nor being forced to give equal consideration to every possibility*'.

At the beginning of a run, Copycat is given an input consisting of three strings of letters. In addition, some initial information regarding the given input must be generated. This is done in the form of attaching elementary descriptors to the various letters in the input strings and the activation of certain nodes deemed relevant for the initial phase. Each letter receives a description regarding its *letter-category* (e.g. the input letter ***a*** is an instance of the letter-category **A**), *object-category* (e.g. the letter a belongs to the object category *letter*, rather than the *group*, or any other, category) and its *string-category* (denoting some salient positions within the sting such as *leftmost*). In addition, each letter is linked to its neighbouring letters (if any) in the string so that Codelets could reach them if needed. No pre-attached bonds or groups are built *a priori*. At the start of each run, the description-type nodes *letter-category* and *string-position* are set to be fully active. This activation is designed to start the programme with some general direction, that is, that the salient positions and the fact that the input letters belong to category letter might be relative to the solution of the given problem. This activation is clamped, however, only for a predetermined number of steps, after which it is released, so that other relevancies could be explored. Notice that such clamping mechanism is not an integral part of Copycat's architecture, but rather an added mechanism that Copycat does not rely upon in any way. When a group is formed by a *group-building Codelet*, the group is defined as a new object, and added to the *workspace*. It then receives its own descriptions just as the input letter had been given descriptions at the start of the run, and can take part (or serve the *Codelets* as substrate for structure building) just like any other object. In addition, if the group's length (i.e. number of letters it consists of) is short enough (reflecting the fact that longer groups are less likely to activate associations consisting of the number of letters), there will be high probability of attaching a *length* description to the group. New descriptions for all objects can always be added by *Codelets*, which will activate their corresponding nodes. These nodes, if fully active will launch new *top-down Codelets* for searching new instances of the same descriptions.

Since Copycat's micro-domain involves string analogies, it is hardwired for a world of three letter-strings, forming a fourth string as a solution. This entails the search for structures in each of the three given strings, as well as correspondences between the initial and target string. As the run progresses, *descriptions* (of objects including groups), *bonds* (between adjacent objects within a string), *groups* (of objects within a string), *correspondences* (between objects in different strings), a *rule* (describing the change from the initial to the modified string) and a *translated-rule* (describing the target-string transformation required for obtaining the answer string) are built. At any moment, structures can be destroyed, and competition between non-compatible structures settles non-coherent organisations.

For the evaluation of the various quality measure of the different structures, Copycat's objects (letter and groups) have a *happiness* measure, while perceptual structures (descriptions, bonds, groups, correspondences, and rules) have a dynamic *strength* measure. The strength of a structure measures its quality. Each structure type has another measure for its strength. For instance, the strength of a *description* is a function of the *conceptual-depth* of the *descriptor*, the *activation* of the *description-type* and the *local-support* of the description. The *local support* is a measure for the number of other description of the same type in the same string. As a result, *relevant*, *deep descriptions* are strong, and so the description-building process is biased towards the use of the same description types already used in the same string. The *happiness* of an object measures how well it fits into the current set of structures and combined with the object's *importance* makes up the object's *salience*. This apparently rich arrangement is designed to allow Copycat to choose salient for a given situation. The object's salience determines how attractive this object is for *Codelets*. The *importance* of an object is a function of the number of *relevant descriptions* it has and the activation levels of the corresponding descriptions are. That is, both the description activation (e.g. *leftmost*) and the activation of the description *type* (e.g. *string position*) count. The *happiness* of an object is a function of the strengths of the structures attached to it. Finally, the *salience* of an object is a function of the object's *importance*, and *unhappiness* (that is, the opposite of its happiness). It follows, then that *unhappy*, *important* objects will attract more attention from future Codelets than *happy* objects of less *importance*. The *temperature* is calculated as a function of the *unhappiness* and *importance* of all objects on the workspace.

As described above, the evaluation and building process of any individual structure is split into a chain of three Codelets. Mitchell summarises this chain as follows:

'First, the ***scout Codelet*** *probabilistically chooses one or more objects on the basis of the* ***relative saliences*** *of the various objects in the workspace…the scout then determines whether its particular type of structure can be attached to the chosen object(s)…If the scout Codelet discovers any reason for building its structure, it places a* ***strength-tester Codelet*** *on the Coderack…When the strength-tester runs, it calculates the strength of the proposed structure, and…decides probabilistically whether to post a* ***builder Codelet****…When the builder Codelet runs, it tries to build the structure, fighting against* ***incompatible*** *already-existing structures if necessary'.*

Before launching the main loop of the programme, the Coderack is filled with the standard population of *Codelets* consisting of *bottom-up scouts*. Equal number of *bond scouts* (searching for inter-string relations between adjacent letters), *replacement finders* (searching for the changes between the initial and modified strings), and *correspondence scouts* (searching for relations between letters in the initial string and the target string) are put on the Coderack. After programme initialisation, the main loop starts iterating, consisting of the following steps:

- **repeat until a translated rule has been found**

    1. **choose a Codelet and remove it from the Coderack**

    2. **run the chosen Codelet**

    3. **if N Codelets have been run then:**

        I. **update the Slipnet**

        II. **add new bottom-up Codelets on the Coderack**

        III. **add posted top-down Codelets on the Coderack**

- **build the answer according to the translated rule**

updating the Slipnet consists of updating the activation of all nodes, by adding the activation from instances discovered in the last N step, spreading activation between neighbouring nodes, calculating activation decay and deciding whether or not to discontinuously fully activate nodes that are more than 50% active.

For a thorough discussion of Copycat's internals, together with evaluation of its performance, see (Mitchell 1993). How does the described architecture relate to the modelling of perceptual processes? What alterations must be done in order to enable perception processes, such as audition, vision, etc. to take place? In the second part of this work, I describe my proposed model, based mainly on the Copycat architecture, and the alterations needed for testing it in the domain of auditory scene analysis.

PART TWO: THE PROPOSED MODEL

5 Requirements for a computational model of perception

Can we avoid future functional problems when designing computational model of perception? There is no doubt that all sorts of design faults will pop up throughout the evolution and lifetime of any model. Still, some basic requirements could be found, that will help to give the model a somewhat natural behaviour. The vast amounts of knowledge gathered by different disciplines could be consulted. Take psychoacoustic data for instance (see chapter 2). Using the results of psychoacoustic experiments for verification purposes only is missing the point. One has got to consider a whole range of such experiments to become aware of the fluid nature of perception. Seeming contradictions among experiments should not be ironed out with epicycles (i.e. tailoring the model to mimic each apparent deviation on its own and ignoring its causes). The fluid nature of perception cannot be bottled by computationally mimicking the results of single experiments while ignoring the underlying apparent functionality. Such functionality only reveals itself when experiments are considered as a whole.

In what follows, I include the requirements I consider vital for the implementation of a computational model of perception. Each section contains requirements arising from a particular perspective. Some requirements (e.g. "allow cues to have different preprocessing algorithms") directly influence the implementation; others (e.g. "allow for inter-cue competition") dictate architecture design considerations in a more subtle manner; others still, (e.g. "see how bees do it") will only help motivating the underlying processes, and understanding the solutions nature has found for similar problems. This by no means renders them less important.

5.1 Psychoacoustics, Neuroanatomy and Computational ASA

Constraints and Requirements from Psychoacoustics and Neuroanatomy

1. Only partial data enters our sensory organs. Individual pieces of such data never correspond to a complete real-world object that is ready for *recognition*. Instead, *pieces of data* are *microscopic* relative to the entities we tend to perceive. Microscopic pieces of data form jumbled up, contradictory *clues* that we use for conjuring up a picture of the outer world. Mirroring the outer world into the cranium using only *clues* necessitates the process of ***perception.*** Consequently, it is the ***interpretation*** of data from multiple features, not the *direct identification* of single objects in the outside world, which cognitive models should aim for.

2. Data from different objects (not to mention all other interfering components in the sensory input, see chapter 2) is mixed up. It looks as if nature has stumbled upon the right way of dealing with such data. It is by looking for pieces of information that best fit together (*relative* to their fit with other pieces) that order could be found in the apparent chaos. In other words, with only simple rules of relating pieces together (e.g. proximity), ***context*** is allowed to take charge, and steer the grouping process in the right direction. It is by *context that* order can be found, not by rigid characteristics of single objects. Single objects break down in any real world input. It follows that the model should be steered by context rather than object definitions.

3. Since this work entails the design requirements for a model of *primitive* perception, that is, with no schema-based capabilities, it should not make use of *a-priori* information regarding the expected nature of incoming sounds. A general model of primitive perception should be able to segregate incoming data into *contextually sensible* groups, which has little to do with pre-knowledge of the arrangement of input objects. The model should therefore ***not*** rely on information such as *"the following input mainly contains the voice of one female and some birds in the background"*.

4. The model should be able to make use of any salient cue for the process of perception. This seemingly obvious requirement is actually more difficult to meet than it seems. The model should allow each cue to have its own behaviour, likes, dislikes, and preprocessing. It should also allow for future extensions by adding new (sorts of) cues. Consequently, the model cannot make any assumptions regarding the nature or the behaviour of cues for its functionality. This can be made possible if the model will make use of *symbolic cues* rather than the (pre-processed) data itself. In such a way, new kinds of cues could be added as symbols with particular parameters sharing existing generic symbolic structure. Alternatively, new modal domains could be added to support entirely new structures. For example, any *local* cue in the spectro-temporal domain could be implemented using a generic symbolic representation of that domain. Other cues, which make use of more global processing such as correlation cues could have different symbolic domains as long as each domain can still exert pressures on others.

5. The implementation of various cues and the search for context rather than object identification, together with the contradictory and incomplete nature of input dictates that: ***Competition*** and ***cooperation*** among perceptually salient cues (as used by human sensory processing) should form the basis for all subsequent emergent activity. Exactly such behaviour is observed in all animal and human perception activities.

6. Grouping principles (e.g. proximity) are found to play key role in perception activities. It is such pressures relate cues together. Other pressures might exist that keep cues apart. In any case, the existence of different types of cues, together with specific pressures results in the cooperative-competitive activity which could bring about the emergence of global grouping. Pressures work on specific features which could be deduced from psychoacoustic experiments, Neuroanatomy, Gestalt psychology and others disciplines. As these *pressures* direct the process of interaction among salient cues, the model should support the definition of different ***pressures*** for different (types of) cues. One should be careful to keep the generic model itself (which supports the *use* of pressures) from the pressure definitions themselves. This might prove to be more difficult than it seems.

7. Additionally, pressures should exist which relate the structures of groups of cues with each other. This can be seen as building additional layers of abstraction on top of the bare cue input. Context plays a significant role in the grouping process while (complex) structures can always be described in alternative ways. So relations among structures take on the form of *analogies* (i.e. analogies in the sense used by *Copycat*, where related structures share a similar *interpretation* of their respective local contexts, see chapter 4). Once such interpretations are made, structures could be treated as meta-cues for pressures working at higher levels to react on. The model should allow for such hierarchies, and should not demand the finalisation of lower-level structures before higher-level pressures are allowed to work. Rather, regrouping at lower levels should still be allowed. Obviously, such regrouping should become more difficult if strong higher-level structures have already been built. Such analogous relation between structures is the key to assigning multiple events (individual chirps) to the same real-world entity (a single bird). Although each chirp is unique (e.g. its frequency, duration, glide, amplitude etc.) it is still analogous in structure (it can be seen to be "the same" in Copycat's terminology). Notice that such analogy is *free* of any *a-priori* knowledge of the specific sound since it makes use of analogies between structures in the given context. This will not result in *recognition*, of course, since that would entail further comparison with some learned data. This supports the idea that infants possess the ability to group sounds together prior to being able to recognise them (see Bregman 1990).

8. For the model to be generic enough (since we do not know in advance what types of pressures we actually need) it should accept any kind of pressures (e.g. grouping or segregating cues, inverting relations between cues, etc.) and direction (e.g. allowing pressures to work within single layers of abstraction, to push about lower-levels (top-down) and higher-levels (bottom-up) etc.). Notice that specialised sub-models could be implemented (e.g. a neural network model trained on separating female from male voices) that will exert grouping pressures on the main model and thereby influence the emergent result. In this sense, the model forces neither pre-processing algorithms nor pressure implementation-paradigms.

9. Statistical methods rely on sufficient differentiation among the range of objects to be recognized. If the entire object repertoire is known in advance, and if every object is unique "enough" for it to be distinguished from the others, then one can use these differences "recognize" a given object. Statistical methods (e.g. based on Markov models) have been designed to cope with the correlations within features of specific objects. These methods, however, are passive in the sense that they rely on predetermined context. Such methods do have some (though limited) success in some areas (e.g. speech recognition), and the theoretic relations of such models to neural network models suggest that such techniques could be incorporated into decentralized computational models of perception.

10. Separating the preprocessing phases from the model itself (which works on the symbolic representation of cues) has the advantage that both real-sounds (sounds fed via preprocessing to the model) and preset symbolic arrangements (fed directly to the model) can be used for testing purposes. This sets the stage for the experimentation phase of preparing the model for a specific domain, where both artificial (symbolic) input and real-world input are required.

11. The model should not be limited by any segregation principles (e.g. only splitting input into foreground and background groups, working with single contexts etc.) This is vital since grouping activities should not be centrally controlled, but rather emerge from the micro-activities of local agents.

5.2 Biology

The rich metaphors available from the natural world inspire one to think of alternative approaches to traditional ones. Not surprisingly, it is all too common to use well-known methodologies simply because they are known territories. One of the reasons I had chosen to look for alternative implementation schemes was my view that conventional implementations cannot cope with some of the constraints listed in this chapter.

I do not suggest that the brain is *literally* an emergent community, nor do I have any proof to support such a claim. When presented with a black box, one can only speculate on the nature of the internal processes that produce the so-called input-output mapping. So many parallels exist, however, between the behaviour of perception mechanisms (both external and the little we know about the internal ones) and those of natural emergent systems that the least we could do is study how natural complex systems work. After all, what we are trying to mimic with a computational model of perception is a natural capability that evolved through the same processes, and possibly uses the same principles as other self-organizing systems in the natural world. It is therefore by studying the properties and build of emergent systems that we could learn how such capabilities are achieved. Since the topic of biological emergent systems cannot be fully explored here, I will only mention a few examples. First, we can learn by observing what desirable properties (e.g. fluid responses to novel problems) emergent systems have. Second, by comparing the constraints of our model with such properties, we can gain insight that can help in the implementation of our model. Finally, solutions to specific problems (e.g. avoiding exponential growth of search activities, or efficient resource management) can be studied.

The emergence phenomenon in biology can be referred to as a '… *process by which a system of interaction subunits acquires qualitatively new properties that cannot be understood as the simple addition of their individual contributions.'* (Camazine 2001, p. 31) In the same book, the authors use a simple example for illustrating how system-level properties arise through the process of emergence in a biological system (for examples of non-biological, i.e. chemical or physical systems, (see Camazine 2001, Prigogine 1984).

'The eggs of Dendroctonus beetles are laid in batches beneath the bark of spruce trees. Larvae hatch from the eggs and feed as a group, side by side, on the phloem tissues just inside the tree bark. Previous studies have shown that the larvae emit an attractive pheromone. In a series of experiments, the larvae were randomly placed on a circular sheet of filter paper…the subsequent positions of the larvae were observed over time. … The experiments demonstrated a simple emergent property – a cluster – in a group where the individuals initially were homogeneously distributed. At a certain density of larvae, the system spontaneously organizes itself.'

The above clustering behaviour is very simple. It is predictable from the positive feedback action between pheromone production and physical displacement towards it. So predictable actually, that one tends to dismiss it as non-interesting, or non-emergent "enough". This attitude is taken by those who reserve the mysterious emergence phenomenon to more "magical" examples, where no direct relations can be found between lower-level activities and the emergent properties. Such attitudes, however, miss the whole point of self-organization. The point is that order (or ordered behaviour) can consistently be made to arise from lower-level non-linear, stochastic activity. Moreover, such global order is not *explicitly* programmed in the system. It is a misconception to point out that the emergent properties are *implicitly* coded in the individual entities. In fact, it is the way in which *implicit behaviour* brings about emergent properties, that makes it so advantageous in nature. The most important advantage seems to be the *fluid* nature of global properties. Even in the simple example above, the clustering behaviour depends on the initial larval density. Change any of the interactions (and in more complex systems there is a much greater number of *independent* pressures of interaction) at the individual level, and you get a different set of properties. If the system is well designed for its function (or as seen from an evolutionary point of view, if the system happened to stumble across a fitter implicit code which enabled it to survive), then its emergent properties can be seen as if they were programmed into the system. The resulting behaviour is flexible; it reacts to changes in its environment; it is not centralised, and it is open to interactions from any source, at any level (such as from individuals from different systems etc.) Such "programming" methods are also effective in the way they direct themselves to achieve desired behaviours. In (Camazine 2001, p. 35), the authors put forth the following scenario:

'*Let us assume that natural selection can tune a particular behavioural parameter to a range of values close to a bifurcation point. Then, small adjustments in the parameter for each individual within a group may induce large changes in the collective properties of the group, thereby endowing the group with a wide range of responses and the ability to switch from one behavioural response to another. Flexibility of this type may operate on a day to day basis, or over a longer time span, such as throughout the seasons.'*

In page 37 of the same source, they write: *'Intuitively, it would seem easier for natural selection to make adjustments in the processes underlying an existing structure than to evolve a fundamentally new structure. It seems likely, therefore, that natural selection generates new adaptive structures and patterns by tuning system parameters in self-organized systems rather than by developing new mechanisms for each new structure'.*

There are endless examples for illustrating the effectiveness of the natural approach for reaching flexible order. On some of the more fascinating (and complex) phenomena, and their implications on the evolutionary processes see (Camazine 2001). Keeping in mind that humans are the product of the very same processes which have brought about all other biological self-organizing systems, it is clear that we should study such systems. The effort of basing computational models of perception on self-organizing mechanisms seems worthwhile when one considers the remarkable parallels between self-organizing and cognitive systems.

5.3 Gestalt

It was Wundt in 1879 who introduced the notion of **structuralism** to psychology. Structuralism meant that behaviour was created by putting together elementary elements. The elements of perception, called *sensations,* were thought to bring about the perception of whole objects by a simple act of "addition". It was apparently the illusion of movement created with a toy stroboscope in 1911 (bought by Psychologist Max Wertheimer, one of the founders of the Gestalt school) that triggered the attack on structuralism (Goldstein 2002). The mechanical stroboscope worked by rapidly alternating two slightly different static pictures and creating an illusion of movement. Wertheimer wondered how structuralism could explain this. The problem was that no in-between pictures existed create "real" sensations of motion (that is, a movement of each *sensation* in the picture). It seemed that the notion of structuralism failed to explain this and other related illusions (collected by the Gestalt group at the University of Frankfurt).

As an alternative explanation to structuralism, the Gestalt group came up with several principles of which most basic one was that the *whole is different that the sum of its parts*. Arnheim (1974, as reproduced in Goldstein 2002) shows two images (see Figure 5-1). As Goldstein puts it, '*our interpretation of whether the horse is rearing back or moving forward depends on the whole display, not just the horse. Thus, the horse in* ***a.*** *appears to be rearing back, but the identical horse in* ***b.*** *appears to be moving forward. The presence of the rider in one case and of the lead horse in the other changes our interpretation of the horse's movement'*. Put in the context and terminology of this chapter, perception is an *interpretation* of *all* the interacting *cues*. Such cues may or may not belong to the same modality, they may or may not come from external sensations but all of them will define the specific *context* in the light of which interpretation takes place.

The distinction between the whole and its constituent elements turned the investigation to search for the mechanisms by which elements brought about the perception of objects. In other words, there had to be some process by which order at higher levels arises from distinct elements (or cues). The Gestalt approach was to propose rules which govern that process of *perceptual organization*. Another example of the critical role context plays is given in Figure 5-2 (Goldstein 2002, photo by R. C. James). The collection of black and white patches organises into a Dalmatian while the constituent elements themselves do not seem to hold enough information (such as basic contours) for forming objects.

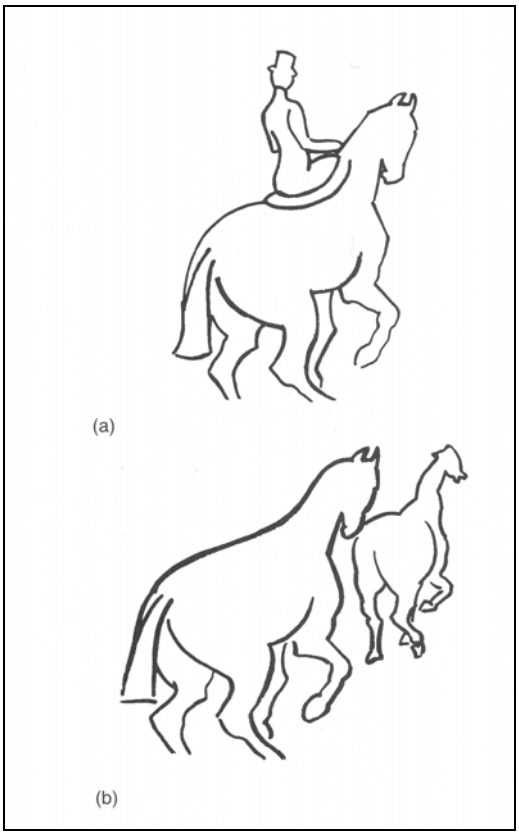

Figure 5-1: (a) A rearing horse. (b) One horse following another. [From Goldstein 2002, p. 148]

Figure 5-2: Some black and white shapes that can become perceptually organized into a Dalmatian (look for the Dalmation head in the center of the picture). [From Goldstein 2002, p. 149]

Some of the *laws of perceptual organization* proposed by the Gestalt school were the

Pragnanz states that the resulting interpretation should form a *good or simple structure*. In other words, the mind will always attempt to *make sense* of *anything*.

Similarity is a pressure which pushes similar properties to be grouped together.

Good continuation, stating that (disconnected) short lines and curves tend to form continuing longer curves if their properties allow it. This law takes care also of the apparent continuation of partly occluded objects.

Proximity is the pressure to group elements which are closer together relative to others.

Common fate describes the tendency of elements to belong together if their behaviour over time (e.g. movement in vision) is similar.

Modern psychologists have rejected the term *law* for describing the above pressures since they do not function as such. Rather, they *push* towards certain directions of interpretation, but it is their interaction with all other pressures which brings about the interpretation as a whole. It is precisely this which relates the Gestalt school to theories of emergent systems. It is important to note that pressures of perceptual organization can be found not only in visual perception but in every sensory perception, and certainly in auditory perception. The reason for this is nearly too obvious to be true but nevertheless, here it is:

Since the constituent elements of any real-world object share parameters and fate, it follows that the best strategy for perceiving the object is to look for what properties the elements ***share****, not for* ***common*** *elements. In other words, it is the context which may direct elements to group together correctly (based on their shared origin) even if they enter the brain in a disordered, jumbled-up state.*

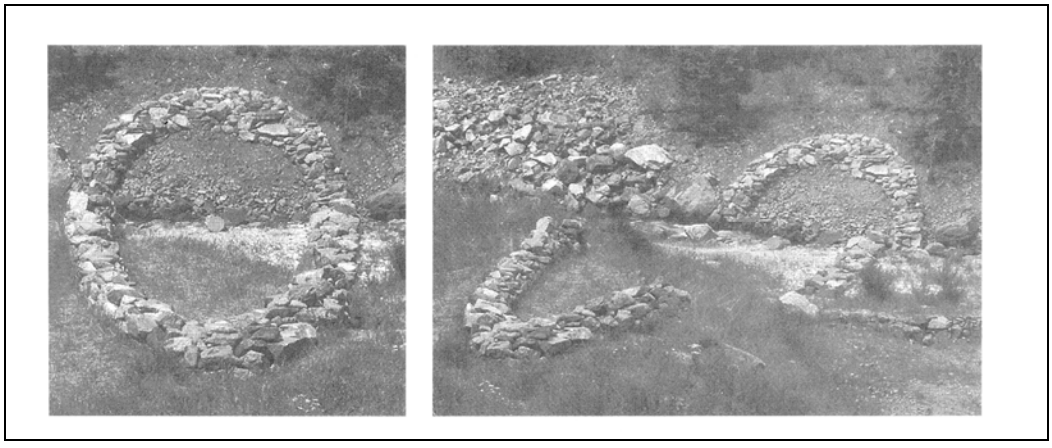

Figure 5-3: An environmental sculpture by Thomas Macaulay. (a) When viewed from exactly the right vantage point (the second-floor balcony of the Blackhawk Mountain School of Art, Black Hawk, Colorado), the stones appear to be arranged in a circle. (b) Viewing the stones from the ground floor reveals a truer picture of their configuration. [From Goldstein 2002, p. 173]

Notice that the problem of perception exists in the first place because we (and all perceiving beings) are never presented with whole objects. Rather, we are presented with a collection of distinct microscopic elements, some of which belong to common objects. In addition, when a given context is (artificially) made to be interpretable in many ways, it could result in a perceptual illusion. In other cases, the best interpretation possible will be used, even if the actual "real world objects" are quite different from the interpreted ones. Figure 5-3 depicts such a case: the context in the left picture is only "half the story", but we have no way of knowing it because the context of the real arrangement was carefully left out by the choice of camera angle. When viewed from a different angle, however, the "real" context is available: same elements, different pressures. The above are related to decentralized implementation mechanisms in the following way:

Constituent elements of real objects produce grouping pressures, and these pressures are anywhere on the line between agreement and disagreement with other pressures. It follows that by letting these pressures actively push each other with no centralized interference, it should be possible that grouping will emerge, which will amount to a **reconstruction** of the shared fate of the elements. In other words, it will result in the grouping of constituent elements into their original objects.

5.4 Bregman's requirements for the 'ideal' casa model

In this section I include Bregman's (Bregman 1990, and Rosenthal & Okuno 1998, ch. 1) constraints for a *casa* (*computational auditory scene analysis*) model (see also chapter 3 on casa in this work). Note that although the terminology is specific to auditory scene analysis, the requirements themselves could (and should) be applied to any other domain of perception.

In (Rosenthal & Okuno 1998, p. 4-5), Bregman writes: '*…I think the easiest part is building mechanisms to extract the cues from the incoming data. It is a good deal harder to figure out how to make them interact properly to converge on a good grouping of the sense data, because this involves choosing an architecture for the system'*. Bregman refers to his view (which I happily support) that the crux of any perception mechanism is the ability of different sensory cues (as well as higher-level structures) to interact. This interaction should lead to the emergence of good structures under certain conditions. To achieve this, Bregman suggests a number of architectural guidelines.

First, the model should provide the basic functionality to allow for such interaction of cues to exist. In this sense there is a separation between the model's architecture, and the definitions of the cues which interact among each other. It still has to be seen whether the line separating the cues (which seem to be more domain specific) from the model (which is hoped to be a more general architecture) can also be the dividing line between domain specific structures and the "general-purpose" engine of perception. But as a constraint, and from a pragmatic point of view (since we still have to use the more general model for experimenting with alternative types and forms of cues), we can start with Bregman's weaker suggestion that '*… the architecture of the system should allow evidence provided by different cues to be added up easily in determining the final organisation'*

Bregman points out the existence of different modes of interaction, such as between sequential and simultaneous integration (that is, competition between two different pressures belonging to different cues) and within single types of pressures. (E.g., when two instances of the same type of pressure, but in different locations compete with each other.) In his words:

'*It is also important that the model exhibit the competition that occurs between forces of sequential and simultaneous integration. For example, harmonicity cues may suggest that an acoustic component belongs to a simultaneous group of partials because they are all related to the same fundamental. At the same time, the fact that*

this component is a smooth continuation of an earlier isolated pure tone may activate a sequential grouping that conflicts with the simultaneous one. Competition and collaboration does not only occur between different acoustic properties in the signal, but also when the same relation occurs at different point in the frequency-by-time field. For example, in frequency-based grouping, two components, X and Y, may group sequentially only because their proximity in frequency to one another is greater than to other tones. However, if a third component, Z, is added, very close to Y in frequency, Y may no longer group with X, preferring, instead, to group with the nearer tone, Z. here we see a competition among the instances of a single relation, frequency proximity, at different places in the frequency-by-time field.'

When designing a general model, it is necessary in my opinion to address this problem more generically. In fact, the examples above should be treated just as such – examples of the true nature of interaction. No law exists that *disallows* specific forms of interaction from occurring, and influencing the flow of structure building. In other words, the model should be designed in such a way, that the interaction capabilities of pressures would not depend on their specific implementation details. In reality, it may prove difficult to achieve such an ideal situation, but starting from scratch, one could at least influence the implementation of both cues and interaction capabilities to ensure the "spontaneous" interaction (that is, with no *a-priori* definitions of allowed forms of interaction) of any implemented pressures which happen to occur in a given context.

This brings about an interesting problem. As long as the interacting pressures have a well-defined target to fight for, interaction can take on the form of a fight between two opposing solutions. For example, if a cue exists, which could only link to one of two possible other cues, and if there exists a single pressure for each of these cues to link to the first one, than the interaction can be implemented in the form a decision-making algorithm, which comes up with a winning pressure. This situation is simple only because the example "pretends" to describe a real "interaction" between the two pressures. In fact, it simply describes a direct fight between two opposing solutions. What separates it from real-world interactions is that it is kept isolated from other types of pressures (that is, pressures that do not directly deal with the specific cues described above). A pressure could exist, for instance, from an entirely different type of cue, which pushes for grouping all cues in a given region together. In this case, since the relevant pressures share no common cues to fight on, it is less than obvious how to implement such capability. This problem is closely related to the so-called *binding* problem in cognitive sciences. In general, it seems that the way to approach this problem is to let all pressures influence not only the cues from which they arise, but also their surrounding areas. Bregman's constraint reads as follows.

'The early analysis of signals would have to yield "fields" of supportive and competitive relations among components of the signal'

Two forms of scalability are mentioned: the first form should make sure that*'…one should be able to incorporate cues other than the ones with which the system was first tested, without changing its architecture'.* The second is for enabling the interaction to take place between any *type* of cues rather than a restricted selection thereof, or as Bregman puts it: '*… that the methods should still work when the problems are scaled up from "toy" problems, in which the constraints on the incoming data are sever, to more complex problems with less constrained data.'*

Finally, Bregman touches the subject of scientific model design. It is not only that the design of a correct architecture is far more important than the choice of cues, but that care has to be taken so that the problem tackled is actually the problem of designing a *model of perception,* and not designing a specific engine for emulating isolated results of psychoacoustic experiments.

'…when the range of experiments is small and the number of computer instructions large, the ability to mimic data is not a convincing proof of the adequacy of the underlying theory. Indeed, in some cases, there is no theory at all underlying the model, but merely a set of computations that reproduce the shape of the data. Such models can be seen merely as capturing some of the formal properties of the data, rather than explaining the phenomena.'

It is precisely this problem which motivates the current work. The goal is not the emulation of specific experiments in the field of auditory scene analysis, nor is it aiming at the solution of auditory perception problems. Rather, it is the investigation of the problem of primitive perception in general as a cognitive activity, and the implementation of a computational model that supports the requirements listed in this chapter. In the very same way, the Copycat project was not aiming at the implementation of its micro-domain. It was designed for the investigation of emergent cognitive behaviours. In the next chapter, I describe what makes Copycat fit many of the requirements listed in this chapter, and why I have subsequently chosen it as the architectural basis for *The Ear's Mind*.

6 Copycat and (auditory) perception

In the previous chapter, I described some of the unique requirements with which any model of perception should comply. Constraints from the fields of Neuroanatomy, psychoacoustics and auditory scene analysis were described along with ones originating in computer sciences; Natural self-organizing systems from the fields of chemistry, biology and evolution were shown to share many of the elusive properties needed for perception. It is my opinion that the Copycat architecture fulfils most of the basic requirements needed for the implementation of a computation model of perception. In this chapter, I briefly clarify my motivation for choosing the Copycat architecture (Hofstadter 1995, Mitchell 1993) as a foundation for The Ear's Mind.

6.1 Copycat's and (auditory) perception

In this section, I critically review Copycat's properties in the light of the previous chapter. It is not my intention to do a one-by-one comparison between Copycat's properties and my requirements, since both are prone to evolve through the development phases. Some requirements could be found over-constraining while some of Copycat's existing characteristics might prove less general than I had hoped. Consequently, I review Copycat as an architectural base without scrutinising single properties. Hofstadter and Mitchell had certainly intended to implement a flexible architecture which supports the mechanisms needed for a computational model of perception. Moreover, since I plan to strip Copycat from its original micro-domain and to make it fit to deal with the fundamental problems of perception, the issue is what capabilities the Copycat architecture **could** support rather than what it supports at the present.

1. *Cognitive models should aim for the **interpretation** of data, not the direct identification of single objects.* Copycat was designed to manipulate microscopic pieces of data rather than data at the object level. It follows that the direct identification of objects is not catered for even if it would seem desirable. Since activity at the object level is emergent, Copycat is implicitly directed towards interpretation. To achieve this goal, Copycat can interpret structures in different ways and allows such interpretations to affect the descriptions of other structures. For example, the ***d*** in the letter string ***abd*** could be *described* as a ***D***, but in other circumstances, as a result of contextual pressures, it might be seen as the ***alphabetic successor of C***. Such built-in interpretational flexibility can have great implications on the emergent structures at higher levels of abstraction.

2. *It is by context that order can be found, not by rigid characteristics of single objects. Single object characteristics break down in any real-world input, and the endless real world combination of object characteristics cannot be rigidly defined. So the model should be steered by context rather than object definitions.* This constraint is partially covered in the above requirement. By allowing micro-pressures to influence and direct the emergence of higher-levels, we ensure both *interpretation* and *context* to play the main roles. Copycat was built on the notion that no single piece of data should be semantically significant. In other words, what gives structures in Copycat *meaning* (in terms of the resulting interpretation) is their underlying context (i.e. their relations to other structures).

3. *The model should not make use of a-priori information regarding the expected nature of incoming sounds.* In Copycat only pressures regarding relationships between pieces of data (i.e. ***successor*** is ***opposite*** of ***predecessor***) or their descriptions (***a*** is a letter) are defined. Although other structures are defined (such as the output template form), they do not direct the emergent behaviour of Copycat in any way.

4. *The model should allow for future extension by adding new (sorts of) cues. Consequently, the model should make no assumptions regarding the nature or the behaviour of cues for its function. It follows that the model should make use of symbolic cues rather than the (pre-processed) data itself.* In Copycat, data is given by defining structures and attaching descriptions to them. In other words, all pieces of data are symbolic, and initially, only basic descriptions are attached (such as that ***a*** is a letter). This situation quickly changes to include the inter-relations between different pieces of data and newly formed groups. As the process continues, new descriptions and relations are attached to data structures (Copycat treats both input data and newly formed groups in the same way). The definition of cues is done by attaching *descriptions* and *relations*, and new one these can be simply added to the repertoire of a specific domain by defining *Codelets* (i.e. pressures) and *links* in the *Slipnet*.

5. ***Competition*** *and* ***cooperation*** *among perceptually salient cues (as used by human sensory processing) should form the basis for all subsequent emergent activity.* This is actually one of the basic ideas behind Copycat. Emergence is carried out by letting individual pressures (each pressure is actually divided into smaller steps) "push each other around" in the workspace. The meaning of *competition* as the antonym of *cooperation* is not defined. Rather, pressures that happen to push a given situation towards a common interpretation can be seen from the outside to be *cooperative*. This stems from the local nature of pressures and ensures that their interaction is indeed free of any predefined bias. It is still possible, however, to define links in the *Slipnet* in such a way that certain pressures will lead to calling others. Doing this might risk disturbing the dominance of context in directing the emergent order.

6. *As pressures direct the process of interaction among salient cues, the model should support the definition of different pressures for different (types of) cues.* See the above point regarding Copycat's support for such definitions. One should be careful to keep the generic model itself (which supports the *use* of pressures) from the pressure definitions themselves. This might prove to be more difficult than it sounds. On stripping Copycat from its original domain and the definitions of new cues and pressures (together with other supporting structures), see chapters 6 and 7.

7. *Additionally, pressures should exist which relate the structures of groups of cues with each other. This can be seen as building additional layers of abstraction on top of the bare cue input.* Copycat certainly supports the definition of groups of elements, and there is nothing implicit in its design to stop it from potentially forming layers upon layers of abstraction (i.e. groups of groups of elements etc.) An explicit limit was set in the way the *Slipnet* is coded to fit Copycat with its original domain. However, it is simple to alter the relevant definitions to allow the model to dynamically build any number of levels. Although Copycat allows the definition of such capabilities, it remains to be seen what sort of definition will best fit a specific domain.

8. *For the model to be generic enough (since we do not know in advance what types of pressures we actually need) it should accept any kind of pressures (e.g. grouping or segregating cues, inverting relations between cues, etc.) and direction (e.g. allowing pressures to work within single layers of abstraction, to push about lower-levels (top-down) and higher-levels (bottom-up) etc.).* Copycat allows us to define any kind of local pressures and cues. One example is to extend Copycat to work with multiple workspaces that could interact with each other.

9. *Statistical methods (e.g. those based on Markov models) have been designed to cope with the correlation within a single object's micro-data. These methods, however, are passive in the sense that they make use of pre-determined object characteristics rather than context. However, models should not rely on statistical methods,*

as they do not make use of the dynamic context of input clues. Since Copycat is built on the interaction of local agents, it surely has no global statistical methods built in to influence its grouping behaviour. Moreover, its architecture makes it quite simple to define a few cues and pressures and start testing the emergent behaviour of the system. It is then possible to alter definitions or add new ones. The fact that defining new cues and pressures leaves the architecture unaltered means that little attention needs to be given already defined cues. This does not mean, however, that fitting Copycat with a new domain is a simple task. It is the design of the domain itself, and the decisions regarding how interactions should take place (e.g. how fights between pressures should be carried out) that proves to be extremely difficult. But Copycat's architecture does allow such definitions to be easily made.

10. *Separating the preprocessing phases from the model itself (which works on the symbolic representation of cues) has the advantage that both real-sounds (sounds fed via preprocessing to the model) and preset symbolic arrangements (fed directly to the model) can be used for testing purposes. This sets the stage for the experimentation phase of preparing the model for a specific domain, where both artificial (symbolic) input and real-world input are required.* It should be clear from the points above that such functionality (if catered for by the designing the specific domain) is available.

11. *The model should not be limited by any segregation principles (e.g. only splitting input into foreground and background groups, working with single contexts etc.) This is vital since grouping activities should not be centrally controlled, but rather emerge from the micro-activities of local agents.* Since Copycat was designed as an emergent system, it has no central control to influence or direct the emerging order. The existing central mechanism takes care that the (theoretically parallel and stochastic) architecture will be able to run on a serial computer architecture (which it does by simulating a parallel environment where *Codelets* behave as if they were active at different speeds at the same time). The ability to work with multiple contexts is not implemented in Copycat. In the original Copycat domain, such an ability is not required since the problem calls for a single analogy (i.e. context) per given problem. Real world domains however, necessitate the use of multiple analogies (or contexts) as different structures are often inter-related by multiple analogies. A real-world sound fragment, for instance, would contain voice utterances (which share context) and sounds of other nature, such as bird songs, or noise (which share their own particular context). Finding a solution for this problem is far from being easy.

From the above discussion (and from chapter 4) it becomes clear that the Copycat architecture was designed to offer such functionality that makes it fit to come with the constraints above. The same goes for constraints suggested by Bregman and the general requirements stemming from the study of emergent biological systems. The Gestalt pressure can be defined as well if the specific domain calls for it (in auditory scene analysis, for instance, it seems that the proximity grouping pressure plays an important role). The rest of this chapter deals with some other properties of the Copycat architecture that make it suitable as a foundation for the implementation of *The Ear's Mind.*

6.2 Additional beneficial characteristics of Copycat

In the previous section, I have argued that Copycat shares much of its characteristics with those of models of emergent behaviour. Moreover, it is my view that the mechanisms of perception are similar to natural complex self-organizing systems. In this section, I include some additional properties.

6.2.1 Extendibility

Copycat's architecture allows for extension in a fairly straightforward way. This is due to the separation between those software modules that define the general functionality of Copycat as an emergent system, and those which are more domain-specific. As good as it sounds, life is not that easy when it comes to the implementation of computational models of perception. The grey area seems to be the detailed definition of the interactions between *Codelets*. Copycat supports the emulation of seemingly independent local *Codelets,* including their ability to build or destroy bridges between (and descriptions of) objects in the *workspace*. All this activity can be easily coded in each of the *Codelets'* software module. General code would then only take care of keeping the engine running (calling *Codelets*, updating the various data structures etc.). Due to such code separation, the system can be very easily extended by writing new modules for new *Codelets* and "plugging them in". This, however, only works when *Codelets* follow a uniform interaction protocol. For example, when two conflicting pressures have to fight each other, there is a need for a code which should be able to choose a winner. Writing such a module seems to depend on the nature of the *Codelets*, so that it too must be altered when new types of *Codelets* are added. Moreover, such "in-between" modules seem to be domain-dependent rather than general.

6.2.2 On search algorithms

If the information needed for finding order in the input resides within the data itself (in the form of context as I assume for my proposed model since I leave out all schema-based knowledge forms and only deal with the domain of primitive perception), then it should be possible, in principle, to find that order by using "traditional" search algorithms. Although it would be rather difficult to implement such functionality with non-distributed search-algorithms, it must be theoretically possible since the information relating the different pieces of data resides in the data itself. If this was impossible, it would mean that no emergent - self-organizing - stochastically based - distributed (add in any additional hype terms) system could achieve it either. So why not do it with traditional search algorithms?

Of course, the long answer consists of this entire work. It is proposed that the mechanisms of perception, being the product of evolution, are just one form of many natural self-organizing systems. All the evidence we have points to the working of competitive local entities. (Or pressures, heuristics, laws or whatever you choose to call them) and not for a consistent central-decision engine. But besides a theoretic possibility, it seems that it is practically much more difficult to implement a centrally controlled search algorithm for such tasks. There are just too many possibilities, too many avenues to explore. To narrow the search down, one has to decide which paths down the search tree are worth exploring and which are not. To do that, one has to explore all of them at least to a certain depth. Besides, every new piece of data would necessitate starting a new search. Additionally, it is not clear how such a search algorithm could be implemented to deal with the multitude of *relations* between the various pieces of data. We are talking about searching for the *implicit order among pieces of data*, not searching for the *existence* of specific pieces (or their quantities). This signals the gap between semantics and context.

Copycat avoids much of the problems of central search algorithms by having a decentralised emergent architecture. Its stochastic nature ensures that less salient, weaker structures will still have some chance to influence the emerging order (and thereby to become stronger and more influential) while important structures (judged by their interconnection in the *workspace*) have more say in directing the process. In addition, Hofstadter introduced the "terraced scan" mechanism which breaks tests into multiple heuristics, starting with shallow (quick) tests, and gradually getting to deeper (more elaborate) tests. A thorough description of the terraced scan paradigm can be found in (Hofstadter 1995). It is, however, the combination of these three

mechanisms, *de-centralised emergence, stochastic-based functionality* and the *terraced scan mechanism* that gives Copycat its capabilities. Obviously, only by implementing a working application can the Copycat architecture be tested for expandability and for coping with real-world problems.

6.2.3 Low-level perception

Some criticism of Copycat's functionality has been discussed in later projects of Hofstadter and his group (Metacat, Tabletop etc.). It is interesting to note that most of this criticism had to do with the inability of Copycat to do things "the way humans do them". Granted, they say, Copycat *does* find good solutions to the given problems, and its solutions *do* share similarities both in the way they are found, and in their nature, but Copycat does not *learn* from its past, and it does not *appreciate* its own answers. In some situations, it can even repeat itself within a single run, without *realising* it. It is my opinion that the domains of primitive perception in general *do not* require any of the above functionality. Such characteristics may actually turn out to be the dividing line between the mechanisms of fundamental (or primitive) perception, and higher-level cognitive processes. None of the metaphors used as motivation for Copycat (including the ant colony, the immune system, and the cytoplasm) could be said to *be aware of their own activity*. They may all get into repeating patterns, and they hardly possess learning abilities. Only by adding memory of past actions could such properties become possible. I believe that for the purposes of primitive perception, there is a need for properties of a different nature (such as multi-context capabilities). If the processes of primitive perception would be able to learn and get out of loops, then nearly all of the visual and auditory illusions and paradoxes would work only for the first few times. Schema-based processes can obviously force learned interpretations, and yet, even while they work (we cannot just switch ourselves off) illusions and paradoxes stay affective enough to puzzle us repeatedly. Copycat was designed both as a model of emergent stochastic self-organizing system, as well as as a model for higher-level analogy making. Strong agents vs. strong structures

There is a significant difference between traditional competitive behaviour (as implemented with heuristics and blackboard systems) and Copycat's. Copycat evaluates the strength of already built structures. Strong structures are harder to break up while weak structures tend to lose fights against stronger ones. Weak structures may become stronger and strong ones may get weaker (for instance, if their interconnecting links are no longer relevant). Heuristics used by other models generally make use either of a scoring system or of a predefined hierarchy to make sure that a winner could be found in conflicting situations. In such cases, it is the traditional solution to define scores for each agent. If linking two objects based on horizontal position is in conflict with an already built vertical link, then one would looks up the stronger heuristic between the two instead of comparing the strength of the built structures with the proposed ones. Actually, a whole spectrum exists between the two approaches. For instance, Copycat uses *conceptual depth* scores (see chapter 4) to give more importance to links which are harder to find (conveying deeper relations). There is nothing wrong with that. One could argue, for instance, that *formants* in auditory domains should carry more weight than other, more superficial properties (such as *synchronicity*). My guess is that complex, well-linked structures should have good chances of winning when competing against an arrangement of just a few dominant links. This follows from the fact that the input consists of microscopic pieces of data, none of which should be thought of having great semantic value on their own.

7 The Ear's Mind: Implementation

So far in this work, I have introduced the field of auditory scene analysis and discussed evidence from psychoacoustics neuroanatomy, cognitive psychology, self-organising biological and chemical systems, and computer science to constrain and direct the design of a computational model of primitive perception. I reviewed some important models in the field of computational auditory scene analysis and critically analysed them in terms of their features, capabilities and architectures. Following that, I have introduced the Copycat programme, its motivation, philosophy and implementation. Finally, I have argued that using architecture such as Copycat's as a basis for such a model support many of the required constraints, and lead to the development of a working emergent model of primitive perception. This chapter is all about implementing the proposed model. Although a complete implementation of a working model including many types of cues, Codelets, preprocessing engines and well-tuned components would take much more work, the first phase of implementation has been completed. After introducing the auditory scene as the chosen domain for the proposed model in section 7.1, and discussing the resemblance between analogies in Copycat's original domain and The Ear's Mind's auditory domain in section 7.2, I review The Ear's Mind's first implementation phase. It takes us from Copycat's source code (written in an outdated Lisp dialect) to a modern common Lisp dialect port in section 7.3, and the preliminary choice of auditory elementary units (called cues in The Ear's Mind) in section 7.4. Finally, section 7.5 covers some implemented capabilities novel to The Ear's Mind:

- *Section 7.5.1. covers the implementation of a time-frequency domain in which The Ear's Mind's activities will take place. More domains may be implemented in the future and made to work with each other*
- *Section 7.5.2. discusses the implementation of the four basic auditory cues used in the first phase of implementation. The Ear's Mind allows for the easy implementation of any other types of cues, and their integration in the emergent soup of primitive perception*
- *In section 7.5.3. I touch the role descriptions play in The Ear's Mind and some of their implementation details*
- *Section 7.5.4 deals with the design and implementation of a socket based graphics server (written in tcl/tk) which enables users to graphically watch both input (e.g. different types of cues) and real-time activity (e.g. landing on cues, sniffing for cues, building and breaking of bonds.) while The Ear's Mind is running. Additionally, the graphics server offers some capabilities such as zooming, hiding specific type of events (e.g. broken bonds) and saving postscript images*
- *Proposed alterations to Copycat's original Slipnet are given in section 7.5.5. As The Ear's Mind grows and more cue types and group types are defined, these changes will be integrated in the Slipnet. For now, however, it is viewed as incomplete. This is in line with the first implementation phase which aims at reaching a point of basic functionality allowing experimentation with preprocessing, cue types, Codelet behaviour to take place*
- *Section 7.5.6. introduces the concept of zones in The Ear's Mind, and their implementation details. Zones are used by all functions which access the time-frequency domain locally, and allow comparisons between local measurements to be made. Different types of zones are implemented*
- *Section 7.5.7 describes the bottom-up scouts and bonds implemented so far. Their implementation allows experimentation to be done. It is planned to start implementation phase two with the definition of top-down, as well as other types of Codelets and related structures*
- *In section 7.5.8 I introduce the concept of families, bunches and mutual support which I believe play a key role in primitive perception. The motivation for introducing such concepts into The Ear's Mind stems from the redundant nature of elementary units of perception*
- *Finally, in section 7.5.9., I discuss the implementation details of a preprocessor for analysing sound fragments. The preprocessor is written in Matlab. It produces an output text file defining the cues it extracts from the input sound. Additionally, the interface between The Ear's Mind and the preprocessor is discussed, which allows different methods of experimentation to be done, and new algorithms to be defined for both existing and future cues*

7.1 Auditory scene analysis as a test domain for the proposed model

Throughout this work, I have described the properties of generic primitive perception. To test these notions, it is necessary to choose a domain and implement it within the proposed model. Only then will it be possible to evaluate the design decisions made and to direct the research in the right path. For this project, I have chosen the domain of primitive auditory scene analysis as testing grounds. In previous chapters, I have outlined the properties of the auditory domain and the parallels it shares with emergent competitive behaviour as can be seen in both biological complex systems and in perceptual mechanisms of mind. Within the auditory domain, I aim at the emulation of the primitive grouping activities of type studied in psychoacoustic experiments. Each sound fragment will go through a preprocessing phase which extracts features of interest. These are fed into *The Ear's Mind* as elementary auditory cues. From there on, any number of experiments, model tuning and extension activities can take place. The typical testing procedure is viewing *The Ear's Mind*'s activity in "real time" (using *The Ear's Mind* graphics server) and watching it as the competitive activities settle down. If sound fragments are used from psychoacoustic experiments, resultant grouping can then be compared with the original published results (as done with humans). Since *The Ear's Mind* aims at emulating primitive perception, it makes no use of schema-based knowledge or processes. This rules-out any form of *a priori* knowledge pertaining a given input as well as any *recognition* capabilities the model might otherwise have. Grouping is reached because of contextual pressures only. It is hoped that analogous structures will form groups indicating the relations among them, yet it is not expected that any such structure would be *identified* as belonging to any specific type of sound (such as a "human voice", "bird", "percussive sound" etc.). Having said that, I am not suggesting that schema based processes should be implemented "outside" (e.g. on top, in parallel, following) the emergent engine. Rather, they should be viewed as learned interpretations working as pressures *within* the emergent soup just like any other pressures. The fact that such interpretations tend to be more abstract than the pressures at the level of sensory input does not make them any different from other pressures in the proposed model. From this perspective, I regard the future implementation of schema-based knowledge *principally* straight forward. The implementation of such capabilities, however, will surely prove more difficult than one initially suspects. Still, it is my view that *The Ear's Mind* could be made to work on problems of primitive auditory scene analysis prior to the implementation of higher-level processes, that is, that primitive scene analysis could be made to work on its own.

7.2 Comparing the domains of Copycat The Ear's Mind

Copycat's architecture supports the emergent building of structures resulting from the interaction among agents. We have already seen evidence to suggest that such activity might lead to the capabilities of primitive perception processes. There is, however, a great difference between Copycat's original domain and the domain of auditory scene analysis. This stems from the nature of Copycat's analogy problems. In the original domain, problems are given in the form of three strings of letters. The aim is to find the analogy between these strings. Only then the alteration from first to second string will be seen in a fitting context to enable Copycat to "do the same" to the third string and come up with an answer. Here, the notion of analogy is the interpretation of the input structures (the input strings) in a way which would reveal the analogy among them. The deeper the analogy discovered by Copycat, the better the answer seems to be. Deeper analogies

make use of more connections within the structures and the connections are less superfluous. The task of finding analogies by means of emergence is hard. By the nature of its domain, Copycat is hard-wired to look for the analogous structures between two given strings. It does this by having special *Codelets* for searching within strings. Other *Codelets* look for correspondences between letters of different strings. Although Copycat's architecture can be implemented without its original domain, there remains the question how to transfer the notion of analogy making to the domain of (auditory) perception. They way I see it, analogy is used by perception in a much more general way then currently thought. It is my belief that the basic principle of primitive perception is coming up with an interpretation of structures for reaching a unified explanation containing as much structure possible. This does not imply the blind binding of features to achieve maximum connectivity. The grouping has to "make sense". This means, that each structure should be interpreted in such a way as to reveal its relation to other structures. When this happens, one can start using the language of perception, and say that analogous structures are "the same". This is what makes two chirps of one bird "the same" even if they differ greatly in details.

But how can we find analogies in a sea of features without knowing where to look for them? It seems that the answer to this question must be somewhere along the following lines. If the only difference between features (sensory input) and groups (built by emergence) is complexity of structure, then the further these groups are from the features, the more *interpretable* they become. This implies that the structure of every group (above a minimum complexity) could be described in many different ways. Take for example a simple case of four cues such as the ones depicted in Figure 7-1.

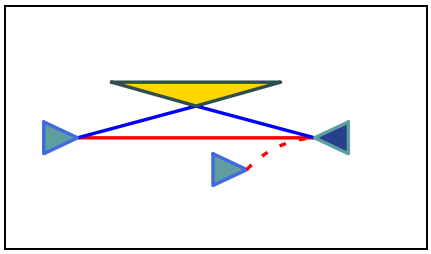

Figure 7-1: A positive temporal derivative cue pt1 (left) attached to both a negative temporal derivative cue nt1 (right) and a negative spectral derivative cue nf1 (top). The additional positive temporal derivative cue pt2 at the bottom has failed to attach itself to the nt1. Even though pt2 is closer to nt1 than the already attached pt1, pt2 has lost a fight against the conflicting bond between pt1 and nt1 for reasons of mutual support from bonds pt1-nf1, and nf1-nt1. The dashed spline indicates a lost fight.

Both cues *pt1* and *pt2* are of the same type. They represent salient points of positive temporal first derivatives in the time-frequency domain. Such points however are not automatically taken in *The Ear's Mind* to be onset points. They only represent points at which a substantial increase in energy occurs in the power spectrogram in the positive temporal direction. Now, say cues *pt1, nf1,* and *nt1* form a group together. New descriptions can now be attached to that group. In this example, it might be appropriate to consider attaching an onset description to cue *pt1*, and define a (positive temporal) boundary edge for the group. In this case, if cue *pt2* would form a bond with this group it may be interpreted as an amplitude modulation point within the group. Of course, other alternative grouping strategies are possible even in this simple example. One such alternative is to bond cues *pt1* and *pt2* together by a temporal extension bond. This would lead to the interpretation of both cues as a single unified boundary edge just as if they had originated in one large cue in the first place. Alternatively, grouping only cues *pt2, nf1,* and *nt2* would leave out cue *pt1* altogether. Just as elementary cues have descriptions attached to them that direct their interaction with other cues (through the activity of Codelets) and the state of concepts in the Slipnet, so will the descriptions attached to groups influence *The Ear's Mind* activity. The only difference between cue descriptions and group descriptions is the interpretative flexibility in

describing groups. Consequently, if groups do not fit each other (or in Copycat terminology, if they are not happy), they could be pushed to be interpreted in alternative ways resulting in greater overall coherence. By allowing cues to form groups, we can use the same mechanics of analogy making (in the copycat sense) to produce coherent interpretations of structures at higher levels of abstraction. This seemingly technical approach opens up the possibility to see different groups as 'the same' even if they differ in detail. Being 'the same' (or 'not the same') now relies on context just as we wanted it to (see for example how the same image can have two different context-sensitive interpretations in Figure 5-1).

7.3 Porting Copycat to a modern dialect

Copycat was implemented in a (now dead) Common Lisp environment using the *Flavors* object-oriented system. This was the main reason why Copycat will not have run on any modern Common Lisp implementation. The modern common Lisp object-definition standard named CLOS (Common Lisp Object System) was adopted in the early 1990s and is currently supported by every common Lisp environment. The Flavors system was developed at the MIT Lisp Machine group in 1979. Back then, the term "Flavor" was a popular jargon word (borrowed from ice cream) for "type" or "kind" at MIT, and was used for denoting the class definition of objects. It supported multiple inheritance, and complex protocols for combining different Flavors together. For more on the history of the Lisp object systems, see (**Norvig** 1992). The first step of getting Copycat to run on modern computer systems was to port Copycat from its original dialect to CLOS and the modern standard of common.

In addition, Copycat made use of a specific graphics environment which provided graphical output and gui. At the time, no standard portable graphics environment existed for Lisp systems. There's still none. Since I have implemented a portable graphics server specifically for *The Ear's Mind*, there was no need of porting Copycat's graphics capabilities. Consequently, I ported Copycat's text-mode functionality only. All graphics functions were removed (or replaced with dummy versions).

Copycat is a large programme: the (commented) source code file size is 568kB containing 15000 lines (divided into separate code files). The programme was written in separate modules. A system-definition file was used to automate the compilation process (including automatic version control). Copycat contained 41 separate files which had to be translated and compiled before I could use it for implementing *The Ear's Mind*. It seemed simpler to port the old Lisp dialect into a modern one rather then translate the whole system into an entirely different language (e.g. Java or C++) for several reasons. First, porting between dialects is much easier and faster since much of the original code would be kept intact. Secondly, Lisp itself by its simultaneous interpreted/compiled nature offers the programmer the ability to test new modules (and general functionality) in a straightforward and interactive way. The third reason was my belief that staying in the language of the original code would not only help future comparisons between the original Copycat code and *The Ear's Mind*, but would actually facilitate the design process of *The Ear's Mind* since I would be able to keep 'thinking in Copycat' while implementing *The Ear's Mind*. Finally, since Lisp code excels in the automated production of Lisp code, it is a somewhat straightforward task to write a compiler in Lisp which can read an old Lisp dialect and produce a Lisp code in the new dialect. This because the input generally shares syntax with both desired output and the common-Lisp reader itself (the Lisp module in charge of parsing Lisp input).

7.3.1 Flavors2clos

For translating the old dialect into the new one, I have written the Lisp function port-ccat, which goes through a list of files containing the Copycat code, and proceeds to translate each file into the new dialect. It then writes the new code in a new file which will (hopefully) run in modern common-Lisp environments. However, this function was designed only to port the Flavors object-oriented system into CLOS. This task was estimated to require the majority of alterations needed to get Copycat to run. Consequently, in some places, manual intervention took place to cope with differences between the dialects in non-object Lisp code. Additionally, solutions had to be found for different standards of inheritance and qualifying mechanisms since not all of Flavors' features are supported in CLOS. Below is the main function for translating the original Copycat into CLOS:

The code (on page 82), first calls function scan-file-flavors. The function goes through the total Copycat code and scans for all defined flavor slots. Each defined slot is registered in an object structure so that the porting engine will be able to decide on the qualifying method it has to use for class definitions (this is due to details pertaining the use of qualified and non-qualified slots in the flavor standard). Once the slot structure has been created, the translator proceeds to process each source-code file on its own by calling function port. Function port calls other functions in turn, which handle file creation and other necessary tasks. Subsequently, each Lisp phrase is translated by function 'translate', which handles the actual porting. The translation from Flavors to CLOS turned out to be tricky at times due to small implementation details of the two dialects. It will suffice to include here a simple example of the original vs. ported code.

Original Flavor code (comments in code were edited out for ease of comparing the dialects):

```
(defflavor workspace
    ((replacement-list nil) proposed-correspondence-array
     correspondence-vector)
    ()
    :gettable-instance-variables
    :settable-instance-variables
    :initable-instance-variables)
```

Ported CLOS code:

```
(defclass workspace nil
 ((replacement-list
       :initarg :replacement-list
       :accessor :replacement-list
       :writer :set-replacement-list
       :initform nil)
  (proposed-correspondence-array
         :initarg :proposed-correspondence-array
         :accessor :proposed-correspondence-array :writer
         :set-proposed-correspondence-array)
  (correspondence-vector
         :initarg  :correspondence-vector
         :accessor :correspondence-vector
         :writer :set-correspondence-vector)))
```

(defun port-ccat ()

```
;; this is the main function for the translation of Copycat. it first ;;
scans all the flavors defined in the code from a global file which ;;
contains the complete code, and then proceeds to translate each   ;; file on
its own.

  (progn
(scan-file-flavors  "c:/home/lisp/orig-ccat/total.l")
    (setq files '(
         ("constants" ".l.altered")
         ("util" ".l.altered")
         ("workspace" ".l")
         ("workspace-structures" ".l")
         ("workspace-strings" ".l.altered")
;changed the 3 setq manually into 3 setf
;due to differences in dialects
         ("workspace-objects" ".l")
         ("initialization" ".l")
         ("run" ".l")
         ("Slipnet-def" ".l.altered")
;inserted &optional plus a dummy parameter in :apply-slippages
;to work with :apply-slippages from descriptions.l.altered
         ("Slipnet-links" ".l")
         ("Slipnet-functions" ".l")
         ("Slipnet-graphics" ".l")
;altered  method :apply-slippages by inserting &optional.
;added a method to the translated file to match nr. of args with ;:other-obj
in correspondences.l.altered
         ("bonds" ".l")
         ("groups" ".l")
         ("concept-mappings" ".l")
         ("correspondences" ".l.altered")
; &optional added to method :other-obj to match :other-obj in
; descriptions.lsp which was altered after translation.
         ("replacements" ".l")
         ("breakers"  ".l")
;ALTERED .LSP FILE    ("Coderack" ".l")
;put the arg BEFORE the instance to comply with arg/instance
;order of all automatically generated  :set- methods
         ("rule" ".l")
         ("answer" ".l")
         ("workspace-object-formulas" ".l")
         ("workspace-structure-formulas" ".l")
         ("formulas" ".l")
         ("graphics-util" ".l")
         ("Coderack-graphics" ".l")
         ("letter-graphics" ".l.altered")
;the following 5 files were altered to match each other's :init-graphics
method's nr. of args. the one with 1 arg (in description-graphics) was made
&optional. the rest got an &optional dummy arg.
         ("description-graphics" ".l.altered")
         ("bond-graphics" ".l.altered")
         ("group-graphics" ".l.altered")
         ("correspondence-graphics" ".l.altered")
         ("replacement-graphics" ".l.altered")
;end of the above mentioned 5 files
         ("rule-graphics" ".l.altered")
         ("workspace-graphics" ".l.altered")
         ("temperature-graphics" ".l")
         ("ccat-bar-graph" ".l.altered")
         ("ccat-menu" ".l")
         ("copystat" ".l")

))
    (loop for item in files
      do (port (car item) (car (cdr item))))
))
```

Following the above defflavor (ported into defclass) are many method definitions which allow (generic) functionality on the defined classes. An example of one of these follows together with the translated version.

Original defmethod code (comments in code were edited out for ease of comparing the dialects):

```
(defmethod (workspace :add-proposed-correspondence) (c)
  (aset (send self :proposed-correspondence-array)
          (send (send c :obj1) :string-number)
          (send (send c :obj2) :string-number)
        (cons c (aref (send self :proposed-correspondence-array)
                      (send (send c :obj1) :string-number)
                      (send (send c :obj2) :string-number)))))
```

Ported CLOS code:

```
(defmethod :add-proposed-correspondence ((self workspace) c)
 (aset (:proposed-correspondence-array self)
          (:string-number (:obj1 c))
          (:string-number (:obj2 c))
       (cons c (aref (:proposed-correspondence-array self)
                      (:string-number (:obj1 c))
                      (:string-number (:obj2 c))))))
```

7.3.2 Miscellaneous porting issues

Several exceptions had to be manually altered. Below is a list of the main issues.

1. The flavor system allows the number of arguments given a method defined on different flavors to vary. This feature is not supported in CLOS. CLOS demands the same number of arguments to be defined for all methods sharing a name. Consequently, I have added dummy parameters to equate the number of arguments in such cases.

2. The few Flavor "send" forms within quotes were not properly caught by the Lisp reader. These were handled manually.

3. A few functions available in the old dialect but not in modern common Lisp (such as function memq) had to be explicitly defined.

4. The Flavor system allowed non-initialized slots, which defaulted to nil. In CLOS, non-initialized slots have no value and produce an error when read. Consequently, the translator produces code to initialize such slots to nil in the new code.

5. A new - compiling system is now used based on the common Lisp "defsystem" module. It allows for automatic version control and loading / compiling of code in a similar manner to the well-known "make" system. By using this system, it was possible to easily add new modules (e.g. specific to *The Ear's Mind*) or alter existing ones.

Following the porting phase, it was possible to run Copycat in text mode and test its functionality. It was now possible to alter Copycat's implementation, add new modules and replace its original domain with the domain of *The Ear's Mind*.

7.4 On the elementary units of auditory perception

The Ear's Mind's architecture was designed to cope with any kind of input cues. This capability is achieved by the abstraction of the type of input used into a set of properties belonging to each input element. In this way, every element has a position on the input domain. In the spectro-temporal domain, each element is defined by several time and frequency anchor points. Although an element has a well-defined position in its domain, its type is still undefined. Consequently, any type of element with a spectro-temporal position can be defined as input. Element types are defined by attaching appropriate descriptions. Such descriptions define the pressures that activate interaction in *The Ear's Mind*. Each input cue must get at least one description to take part in the grouping activity of *The Ear's Mind*. Cues with no descriptions are not chosen by Codelets for grouping purposes. These initial descriptions (indicating sensory feature types) are not interpretable. They will stay attached to the element for the length of the run. New interpretable descriptions can be attached throughout the course of the run to both single elements and groups.

As described in (chapter on 3), there is a great diversity in the choice of the so-called primitive unit of perception – that elusive minimal piece of evidence – the smallest building block of input used in perception. Some argue for acoustic elements, i.e. those that exist in the original input, such as signal power or intensity changes, while others opt for the processed signal-domains found in sensory anatomy, such as auditory nerve spikes, or the more elusive cross-correlation alternatives. There exists a tendency to implement elaborate preprocessing phases which break the input signal into its constituent primitive units of perception and then hand these over to an engine which is supposed to group these together into perceptually meaningful chunks. However, the choice of the dividing line between the two processes is crucial. Two popular approached exist. The first is to work with such elements which resemble the primitive units used as in psychoacoustic experiments. The second – to work with more global arrangements which compute correlations within the input signal. Both of these approaches are using blind decision-mechanisms for building the primitive building blocks. There is indeed very little choice in the matter, since a more elaborate, dynamic model needs input elements to work on. Anatomical evidence (see chapter 2) supports that view, and shows that the mechanisms of the cochlea are far more blind and passive in comparison with later processing phases.

To me, the crux of the problem lies in the notion of interpretation. Since I view the entire process of perception as interpretative, I argue that all preprocessing blind mechanisms be confined only to tasks of feature extraction. Of course, in practice, such an ideal is never achievable, but the crucial point is to leave as much blind interpretation as possible out of the preprocessing phases. Take for example abrupt temporal intensity changes in the input signal. The tendency among researchers is to treat such points as onsets or offsets. This stems from psychoacoustic experiments showing the ear's sensitivity to such intensity changes, and its tendency to *interpret* such points as boundary points of auditory events. The fact is, however, that such abrupt intensity changes may or may not indicate the starting (or ending) point of an acoustic event. Surely, in traditional psychoacoustic experiments, such intensity changes are indeed the onsets and offsets, but needless to say, it is by design. Once a preprocessing engine

makes the decision to name such a point an onset, there is no way back. (unless, as some argue, the perceiving engine is so smart that it could correct the preprocessing engine, rendering it useless.) Subsequently, I believe that onsets and offsets should be used as interpretative descriptions, and should be only assigned by context. It follows, that onsets and offsets should not be taken to be properties of the primitive units of perception. There is every reason to believe, however, that temporal as well as spectral abrupt changes in intensity play a key role in auditory perception.

For practical reasons, I have decided to implement intensity changes in the first implementation phase of *The Ear's Mind*. I only regard such points as salient auditory events and not as starting or ending points. Further, it should be clear by now, that *The Ear's Mind* allows for the definitions of other types of elements and nothing in its architecture relies on input specifics. Choosing to implement intensity changes first allows for simpler preprocessing algorithms. This choice, however, is motivated only by the need to get things started, and does not imply that the use of intensity derivatives only is sufficient for the purposes of auditory perception. It is my view that the quest for the ideal primitive element of auditory perception is futile. Instead it should be the result of the interaction between many different types of cues in a given context which will bring about primitive perception. It starts at the feature level itself, denoted as *noticed cues*. (The word *cue* implies that features are only treated as evidence for salience and do not hold any static significance in themselves. The word *noticed* is used since I allow in *The Ear's Mind* for cues to go unnoticed since both search-zone and resolution do not have to be exhaustive. In analogy with the working of the eye which can explore both new areas as well as new details, *The Ear's Mind* would be able to request more cues if needed). The input signal is searched for the existence of features, which are then used by dynamic grouping pressures to form perceptually important arrangements. These arrangements may produce groups of features. Such groups will form auditory events which will accepts new descriptions. Such descriptions will in turn bring about new pressures and carry the grouping process further.

A cue can hardly qualify for boundary status without consulting the surrounding context first. This is valid even if the derivative is greater than all the others in its zone. This is because of the following two reasons:

- We do not know what zone-zoom is appropriate, and different zone dimensions will be needed in different situations.
- Taking the greatest derivative within a zone does not guarantee that no greater derivative lies just outside that zone.

The solution is to qualify a boundary by other cues. For instance, a positive temporal cue might get support from a spectral cue close by. Of course, already-grouped cues (e.g. a left boundary grouped with a top and bottom boundaries) could give strong support to a salient negative-going temporal derivative in the right place. In that case, it could join their group. Implementation of such functionality could be done in the following way. Codelets may activate cues to produce a broadcast signal, a 'scent', which signifies their readiness to bond with other cues. Scents could have their strength weighted by their distance to the seeking cue, and their salience. As time passes, scent may be made to weaken. So when a Codelet is sitting on a temporal-positive-going derivative and looking for a negative-going one, he can sniff for negative derivatives and get signals from a few; some far-off but yet strong, others nearby, yet quite faint. This affects the way the Codelet goes about its work. Instead of searching by salience or by distances, he stochastically picks 'a strong' signal, which favours the following chances:

1. Its not too far away, though it might still be situated beyond some weaker, closer cues
2. Its important enough
3. It has not been too long since it started to broadcast scent (since salient cues start broadcasting first, they have greater chance of dominating structure building).

4. Weakening with time has the advantage of avoiding the same few salient cues if they do not result in strong structures.

At any time, already grouped cues should post their characteristics so that other cues would be able to join in, or even challenge one of the already incorporated cues if they 'think' that they are fitter for its position.

7.5 Implementation of The Ear's Mind

In each of the following sub-sections, I discuss the implementation details of one or more related features of *The Ear's Mind*. The following features are handled:

- *the time-frequency domain*
- *auditory cues*
- *descriptions*
- *the graphics server*
- *the Slipnet*
- *zones*
- *scouts and bonds*
- *families, bunches and mutual support*
- *the preprocessor*

For a graphical representation of the global structure of The Ear's Mind, see figures Figure 7-2 *(preprocessor),* Figure 7-3 *(main module), and* Figure 7-4 *(Graphics server) below.*

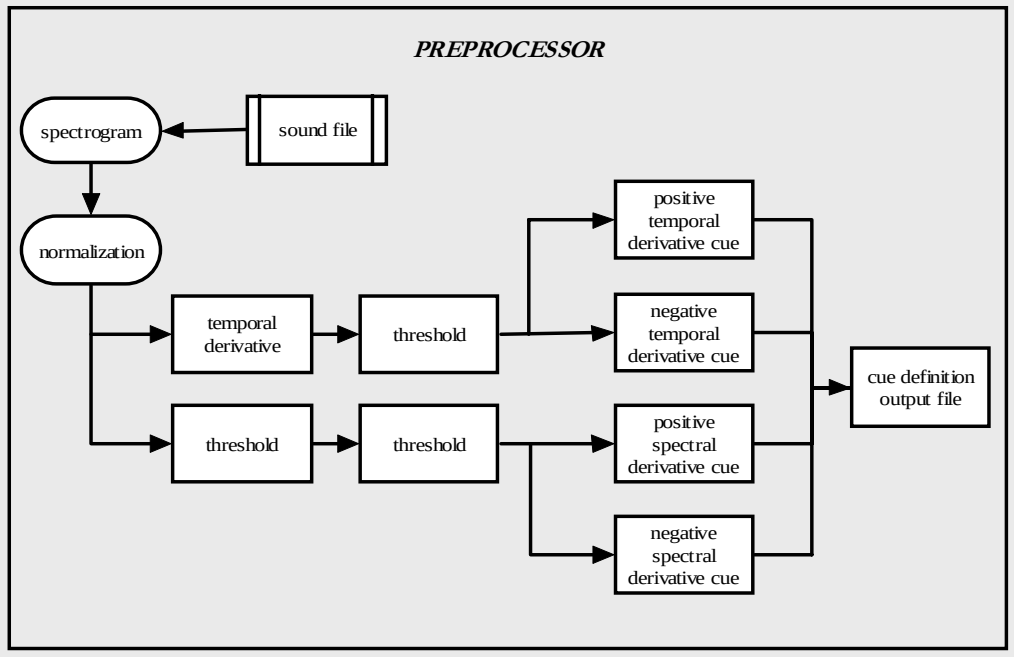

Figure 7-2: block diagram of The Ear's Mind's preprocessor (written in Matlab)

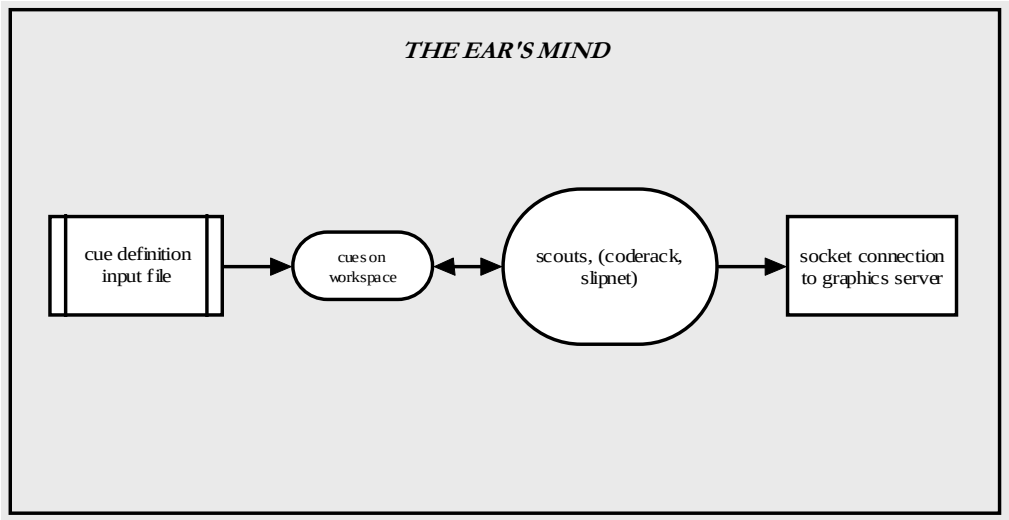

Figure 7-3: block diagram of The Ear's Mind's main module (written in common Lisp)

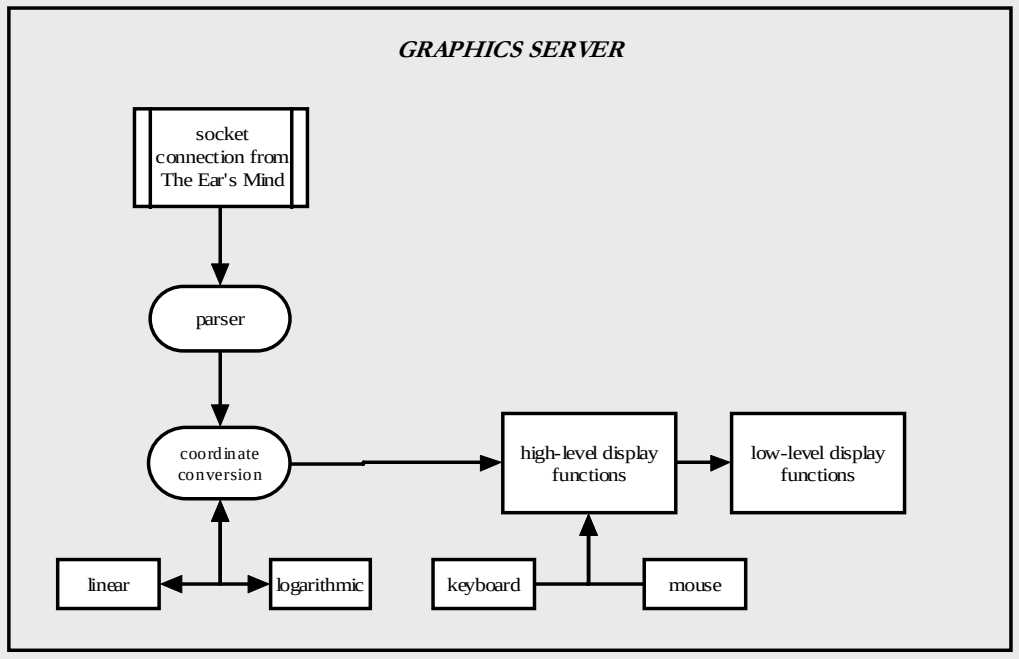

Figure 7-4: block diagram of The Ear's Mind's Graphics server (written in tcl/tk)

7.5.1 Implementing a time-frequency domain

As *The Ear's Mind* is strongly bound to time and frequency dimensions, a substrate was required for supporting the two dimensional indexing of all cues in *The Ear's Mind*. The basic structure of the implemented time-frequency domain is described below.

We start by implementing a tf-vector with the following slots:

- *Category* can be either time-vector or freq-vector, indicating the type of tf-vector it is
- *Stamp* contains either a time or a frequency value. Time values are given in seconds and freq values in Hertz. The category slot (either time-vector or freq-vector) determines the type of the stamp value.
- *Next* points (using a linked list) to the vector of the same category type, whose stamp is greater than the stamp value of the current vector, but smaller than all other defined vectors. A nil indicates that the current vector has the largest stamp of its category.
- *Previous* points (using a linked list) to the vector of the same category type, whose stamp is smaller than the stamp value of the current vector, but greater than all other defined vectors. A nil indicates that the current vector has the smallest stamp of its category
- *Vector* is a list containing all the registered tf-points, which share the stamp of the current ft-vector instance.

```
(defclass tf-vector nil
((category . . . ) ;;either time-vector or freq-vector
 (stamp    . . . ) ;;the stamp, i.e. value of either time or freq.
                   ;;of the vector and all its ft-points
 (next     . . . ) ;;points to the next instance of tf-vector of the
                   ;;same category
 (previous . . . ) ;;points to the previous instance of tf-vector of
                   ;;the same category
 (vector   . . . ) ;;a vector containing all the tf-points sharing
                   ;;the same stamp of the current tf-vector
                   ;;instance.
))
```

Tf-vectors are defined both for time stamps as well as for freq stamps for each new time or freq value. A tf-point is defined with both time and freq stamps and belongs to an object (such as a cue). Actually, a cue in the current implementation will have four such points, corresponding to the beginning and end-points in both time and frequency directions. These points define the cue boundaries and should not be confused with the boundary of a perceived auditory event. Every tf-point is included in the vector slots of both the ft-vector of time category and the tf-vector of frequency category which share their corresponding stamps.

The tf-point class contains the following slots:

- *Category* indicates the type of tf-point it is. This slot is meaningful to functions searching for specific types of points. For instance, one might need to search for the existence of a top-left corner of any object within a given tf-zone. This slot can have the following values: bottom-right (the point of lowest frequency, greatest time of the object to which this tf-point belongs), bottom-left (lowest frequency, smallest time stamp), top-right (greatest frequency, greatest time), top-left (greatest freq., smallest time), center (used for graphics and calculated as follows: frequency stamp is (max object freq – min object freq)/2 , time stamp (max object time – min object time) / 2), front (used for graphics: centre freq and greatest time), and format (denoting a formatting point enabling more efficient searches). Currently, at init time, format points are defined forming a grid of 1000 Hz. and 1 second intervals.
- *Time-stamp* the value in seconds of the temporal position of this tf-point on the tf-domain.
- *Freq-stamp* the value in Hz. of the spectral position of this tf-point on the tf-domain.
- *Time-vector* points to the tf-vector of type time-vector in which this tf-point is registered.
- *Freq-vector* points to the tf-vector of type freq-vector in which this tf-point is registered.
- *Cue* points to the object to which this tf-point belongs.

```
(defclass tf-point nil
;stands for time-frequency-point.
;used in class coordinates which is in turn used in all objects that ;have
coordinates. for now these are all the workspace-objects class.
;the stamp slots denote the actual time and freq values.
;the vectors are instances of vectors which are defined by the tf-;vector
class. all tf points sharing the same time stamp are included ;in the same
time-vector instance likewise for the freq-vector
  ((category . . .)
   ;;either a bottom-right , bottom-left, top-right , top-left,
   ;;center, front, format.
   (time-stamp . . .) ;;the time-value of this tf-point in seconds
   (freq-stamp . . .) ;;the freq-value of this tf-point in Hz.
   (time-vector . . .);;points to the tf-vector of time category
                      ;;to which this tf-point belongs
   (freq-vector . . .);;points to the tf-vector of freq category
                      ;;to which this tf-point belongs.
   (cue . . .)        ;;points to the instance to which this tf-point
                      ;;belongs.
```

The tf-domain is initialised by calling init-tf-vectors, which defines the first time- and frequency-vectors with zero stamps and creates formatting vectors every 1000 Hz and one second:

```
(defun init-tf-vectors ()
;; initialize the two global vectors for time and frequency axes.
;;these are used for time and freq stamps and ordered linking of
;;objects.
  (setq *em-time-vector*
   (make-instance 'tf-vector
              :category 'time-vector
              :stamp 0
              :next nil
              :previous  nil
              :vector nil))
  (setq *em-freq-vector*
   (make-instance 'tf-vector
              :category 'freq-vector
              :stamp 0
              :next nil
              :previous nil
              :vector nil))
;;format the time-freq zone for faster searches and inits.
;;for each second, and every 1000 Hz put a tf-point with
;;category format. this can be used for searches by starting the
;;search at the appropriate truncated time and freq to the lower
;;coordinates.
 (setq *time-array-interval* 0.5) ;;interval in seconds of the time
                                  ;;array
 (setq *freq-array-interval* 100) ;;interval in Hz of the freq array
 (setq *time-format-array* (make-array 20 :initial-contents
   (loop for time from 0.0 to 9.5 by 0.5 collect
     (get-tf-vector time *em-time-vector* 'time-vector '?))))
 (setq *freq-format-array* (make-array 24 :initial-contents
   (loop for freq from 0.0 to 5750.0 by 250 collect
     (get-tf-vector freq *em-freq-vector* 'freq-vector '?))))
 (format t "*em-freq-vector* and *em-time-vector* initialised and
    formatted"))
```

Several methods and functions were defined on the above class. The main ones are:

```
(defun make-tf-point (point category cue &aux tformat fformat time freq
tf-point)
;;defines a new tf-point instance with the given time (car point)and
;;freq (cadr point), category and cue object
  (setq time (car point)) ;; set time
  (setq freq (cadr point)) ;; set freq
;; define a new tf-point
  (setq tf-point
   (make-instance 'tf-point
              :category category
              :time-stamp time
              :freq-stamp freq
              :cue cue))
;;find the closest formatting freq- and time- vectors to this point
  (setq tformat (get-tf-vector (floor time 1.0)
                 (floor-time-vector time) 'time-vector '?))
  (setq fformat (get-tf-vector (* 250 (floor freq 250))
                 (floor-freq-vector freq) 'freq-vector '?))
 ;;find the time-vector to which this tf-point should belong. If none
 ;;exists, then create one.
 (:set-time-vector
    (get-tf-vector time tformat 'time-vector '?) tf-point)
 ;;push this tf-point onto vector in the matching time-vector
 (push tf-point (:vector (:time-vector tf-point)))
;;find the freq-vector to which this tf-point should belong. If none
;;exists, then create one.
(:set-freq-vector
    (get-tf-vector freq fformat 'freq-vector '?) tf-point)
;;push this tf-point onto vector in the matching freq-vector
 (push tf-point (:vector (:freq-vector tf-point)))
;;return the tf-point
 tf-point )


(defun get-tf-vector (stamp vec category above &aux end start)
  ;;returns the vec (given vector) stamp vector which corresponds to
  ;;the stamp value. if such a vector does not exist than one will be
  ;;created and attached in the appropriate location.


(defun find-tf-vector (stamp vec category above &aux end start)
  ;;returns the stamp (time or freq ) vector which corresponds to the
  ;;(time or freq) stamp value. if such a vector does not exist ;;return
  nil.


(defun find-tf-points (min max vec-category point-category &aux start-
vec)
  ;;starting with the vector with the smallest stamp equal or greater
  ;;to min, and ending with the vector with the greatest stamp ;;smaller
  of equal to max, make a flat list of all the instances ;;whose tf-
  points of point-category belong to.
```

It is possible to define new tf-points and to register them in the corresponding time and frequency vectors by using the above definitions in combination with functions described in the following sections. It is also possible to search within a given time-frequency zone for the existence of tf-points of specific types (such as bottom-left, or center) and from there, to get to the objects to which these points belong. This lays down the functional substrate for all time and frequency related activities in *The Ear's Mind*.

7.5.2 Implementing Auditory Cues

I have described the choice for primitive elementary units of auditory perception in section 7.4. Consequently, I have chosen to implement four basic types of units for the initial testing phases of *The Ear's Mind*. These are:

- Positive temporal first derivative
- Negative temporal first derivative
- Positive spectral first derivative
- Negative temporal first derivative

We start by defining the top object class from which all cue classes inherit:

```
(defclass workspace-object nil (
  ;;slots for the four boundary coordinate tf-points (for details,
  ;;see definitions of tf-points)
 (bottom-right . . .)
 (bottom-left  . . .)
 (top-right    . . .)
 (top-left     . . .)
 (center       . . .)
   ;;middle points of freq and time stamps. instance of tf-point. For
   ;;graphics purposes
 (front        . . .)
   ;;freq and time coordinates of the front middle point of a cue.
   ;;For graphics purposes. instance of tf-point
 (delta-t      . . .)
 (delta-f      . . .)
   ;;contain time width (max-time – min-time) and freq width
   ;;(max-freq – min-freq) respectively
 (level        . . .)
   ;;indicating levels of abstraction. Input cues are at level 1,
   ;;groups of cues at level 2, etc. currently only level 1 is
   ;;implemented
 (bunch        . . .)
   ;;bunch is the bunch instance this object belongs to. (for
   ;;details, see bunches)
 (outgoing-bonds . . .)
 (incoming-bonds . . .)
 (proposed-bonds . . .)
   ;;lists of all the outgoing, incoming, and  proposed bonds
   ;;respectively (see bonds)
 (tcl-handel     . . .)
   ;;graphics handle of this object. The handle is returned by the
   ;;graphics server at the creation time of the graphics object,
   ;;and is used by The Ear's Mind in all subsequent graphics
   ;;operations (see graphics server)
 (tcl-zone-handel . . .)
   ;;same as above, but referring to the current graphics zone in
   ;;which this object resides.
 (attended?      . . .)
   ;;a predicate returning true if The Ear's Mind has noticed this
   ;;cue and might therefore use it for interpreting the auditory-
   ;;scene. it can be pre-set before the run for some cues to give a
   ;;starting point, or by bottom-up or top-down scouts which seek
   ;;new cues. other than these scouts, The Ear's Mind has no access
   ;;to unattended cues. in addition, and only in the form of a
   ;;speculative future use, The Ear's Mind  could decide to
   ;;ignore certain cues in its scene-analysis and turn their
   ;;attanted states to off.
 (pname  . . .) ;;is the print name such as 'cue or 'group, but could
   ;;also be used for the  purpose of analysis, by giving specific
   ;;names to cues, such as "second harmonic"
   (independent-descriptions . . .)
   ;;a list of all description objects that are currently attached to
   ;;this object. Independent descriptions can be attached to an
   ;;object based on evidence from that object only. That is, the
   ;;description is independent from the existence and attributes of
   ;;any other object.
   ))
```

```
(defclass cue (workspace-object) nil) ;;inherits from workspace-;;object,
see above

(defun make-cue (name attend bl br tl tr &key in-desc &aux cue
description)
  ;;make a new instance of the cue class.
  ;; bl br tl and tr are each a list of (time freq) coordinates
  ;;(t for top, b for bottom, l and r for left and right)
  (setq cue (make-instance 'cue
             :pname   (string-downcase name)
             :attended? attend
             :level 1))
  (:set-outgoing-bonds (make-hash-table :size 2) cue)
  (:set-incoming-bonds (make-hash-table :size 2) cue)
  (:set-bunch (make-hash-table :size 2) cue)
  (:set-bottom-right (make-tf-point br 'bottom-right cue ) cue)
  (:set-bottom-left (make-tf-point bl 'bottom-left cue) cue)
  (:set-top-right (make-tf-point tr 'top-right cue) cue)
  (:set-top-left (make-tf-point tl 'top-left cue) cue)
  (set-deltas cue )
  (:set-center
     (make-tf-point
       (list
         (/ (+ (:max-time cue) (:min-time cue)) 2)
          (/ (+ (:max-freq cue) (:min-freq cue)) 2)) 'center cue) cue)
;;if there is a description then set it; the front point will be set ;;by
the function which called make-cue. otherwise, set the front ;;point to
the center. front and center points are mainly for ;;graphics.
  (if in-desc
      (add-independent-description cue in-desc)
      (:set-front (:center cue) cue))
  (workspace-add-object cue 1)
;;return cue
  cue)

(defun make-time-cue (name low-freq high-freq start-time &key (width 0.04)
(desc nil) &aux cue)
 ;;creates an attended rectangular temporal derivative cue with
 ;;the given low and high frequency in Hz, start time and width
 ;;(defaulting to 0.04 sec.) in seconds, and an independent
 ;;description if one is given.
  (if* desc then
     (setq cue
        (make-cue name 't (list start-time low-freq)
              (list (+ width start-time) low-freq)
               (list start-time high-freq)
              (list (+ width start-time) high-freq)
               :in-desc (if (eq desc 'pos)
                            plato-positive-temporal-derivative
                            plato-negative-temporal-derivative)))
     (:set-front (make-tf-point (list
       (if (eq desc 'pos) (:max-time cue) (:min-time cue))
       (/ (+ (:max-freq cue) (:min-freq cue)) 2)) 'front cue) cue)
    else
       (make-cue name 't (list start-time low-freq)
              (list (+ width start-time) low-freq)
               (list start-time high-freq)
              (list (+ width start-time) high-freq))))

(defun make-freq-cue (name low-time high-time start-freq
    &key (width 35) (desc nil) &aux cue)
;;see above, but creates an attended rectangular spectral derivative ;;cue
with the given low and high time stamps in sec. and start freq. ;;and
width (defaulting to 35 Hz.) in Hz., and an independent ;;description if
one is given.
. . .)
```

```
(defmethod :max-freq ((self cue))
. . .
;;returns the highest frequency that this cue touches.)

(defmethod :min-freq ((self cue))
. . .
;;returns the lowest frequency that this cue touches.

(defmethod :max-time ((self cue))
. . .
;;returns the largest time stamp that this cue touches.

(defmethod :min-time ((self cue))
. . .
;;returns the smallest time stamp that cue touches.

(defun owns-in-desc? (in-desc cue)
;; return the independent description object of the given type
;;in-desc if one exists, otherwise nil. in-desc should be a string
;;such as denoting the name of description such as
;;"PLATO-POSITIVE-SPECTRAL-DERIVATIVE"
(if (:independent-descriptions cue)
    (loop for desc in (:independent-descriptions cue)
     do when (equal in-desc (:pname (:descriptor desc))) return desc)
  nil))

(defun cues-of-in-desc (in-desc cues)
;;returns a list of cues which have the given in-desc from cues (a
;;list of cue objects)
 (loop for cue in cues do when (owns-in-desc? in-desc cue) collect cue))

(defmethod :all-bonds ((self workspace-object))
;;returns a list of all bonds attached to this object
;;by appending the incoming and outgoing bonds.
;;this methods works on cue by inheritance.
 (append (:incoming-bonds self) (:outgoing-bonds self)))
```

Methods were implemented for accessing and computing various attributes relating to the cue class. Some of these are sketched below:

These units should signal the existence of salient pieces of evidence which have been shown to be of significance in audition. Notice, however, that the existence of such a cue, even when the cue is strong (or even stronger than any other cue of its type within a given zone) does NOT imply a *boundary point* of a perceived auditory event. These cues are taken as is, that is, as evidence that the energy levels have changed in a given location and direction. It is by the process of the different mechanisms of *The Ear's Mind* that cues acquire meaning through context. A boundary point signifying either temporal of spectral beginning or end of an auditory event (e.g. tone) is seen as an interpretation of many (types of) salient cues. For this reason, I have chosen not to use the common names, *onset* and *offset*, for describing such points.

For the creation of test cues, several functions were implemented. Some of these are:

```
(defun make-tone (name start-time end-time start-freq end-freq &optional
(time-width 0.02))
;;create a simple tone consisting of four cues which form a rectangle
;;with appropriate independent-descriptions. The simple tone is built
;;by defining its temporal and spectral positive and negative
;;boundaries given a name, and the start and end values of time (in
;;seconds)and frequency (in Hz.). the tone itself does not exist in
;;fact. That is, no tone object is defined, but rather four objects
;;are defined and The Ear's Mind does not in any way know that these
;;objects are in any way related.
;;
;;return the list of the four objects:
  (list
    ;;make a frequency cue with the given position and dimensions
    ;;with an attached description called
    ;;plato-positive-spectral-derivative
   (make-freq-cue (format nil "~aLow-freq" name) start-time end-time
start-freq :width 30 :desc 'pos)
    ;;make a frequency cue with the given position and dimensions
    ;;with an attached description called
    ;;plato-negative-spectral-derivative


(defun (make-freq-cue (format nil "~aHigh-freq" name) start-time end-time
(- end-freq 30):width 30 :desc 'neg)
    ;;make a temporal cue with the given position and dimensions
    ;;with an attached description called
    ;;plato-positive-temporal-derivative
   (make-time-cue (format nil "~aLow-time" name) start-freq end-freq
(- start-time time-width) :desc 'pos)
    ;;make a temporal cue with the given position and dimensions
    ;;with an attached description called
    ;;plato-negative-temporal-derivative
   (make-time-cue (format nil "~aHigh-time" name) start-freq end-freq
(+ end-time time-width) :desc 'neg)))
```

The line `(make-tone 'A 2 3 440 520)` for example will produce the image below (see Figure 7-5).

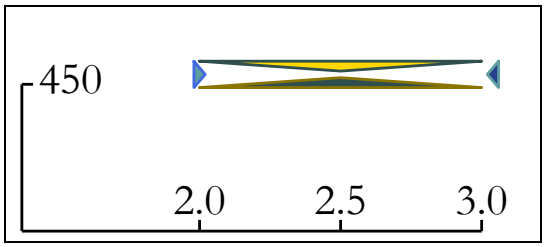

Figure 7-5: creating a simple tone by calling `(make-tone 'A 2 3 440 520)`

```
(defun make-complex (name start-time end-time fund-freq nr-harm
&optional (freq-width 100))
;;create a set of boundary cues consisting of a set of tones.
;;each tone is created with the above make-tone function.
;;the tones are harmonically related and synchronous so that only
;;start-time, end-time (in seconds), the fundamental frequency fund-
;;freq, and the required number of harmonics, nr-harm are given. The
;;optional parameter freq-width can be given (in Hz.) or defaults to
;;100 Hz.
(loop for n from 1 to nr-harm
      collect (make-tone (format nil "~a-~aHz-" name (* n fund-freq))
start-time end-time (* fund-freq n)
          (+ (* fund-freq n) freq-width))))
```

Example uses of implementation for creating single cues, simple tones and complex harmonic tones are:

- Creating a single cue with description: positive temporal derivative
- Creating a single cue with description: positive spectral derivative
- Creating a simple tone with function make-tone
- Creating a complex tone with function make-complex

`(setq complex-a (make-complex 'complex-a 1 2.5 440  4))` will produce Figure 7-6 below:

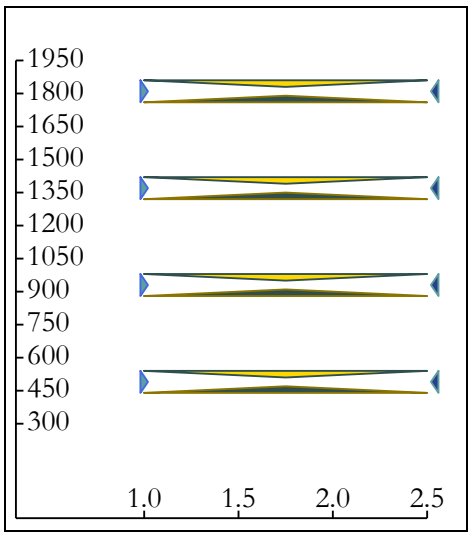

Figure 7-6: ***`(setq complex-a (make-complex 'complex-a 1 2.5 440  4))`*** *produces the above 16 cues making up a complex harmonic tone.*

7.5.3 Descriptions

Descriptions are attached to cues and other higher-level objects (e.g. groups). A cue with no descriptions will not have any effect on the programme, since Codelets work on the attributes of cues. Descriptions are in fact the simplest attributes a cue can have. Later on, higher-level descriptions and relations from the Slipnet will influence the results. The lowest-level descriptions can be thought of being equivalent to the 'is-a' (as in 'an apple *is a* fruit') in traditional artificial intelligence implementations. In *The Ear's Mind*, two kinds of descriptions are defined: Independent descriptions are those descriptions that need no other evidence except from the evidence local to a cue for attaching to (or detaching from) that cue. Such descriptions are simpler in their behaviour, since they do not depend on the existence and activity of other surrounding cues. In Copycat, the existence of independent descriptions (called descriptions in Copycat) was enough due to the relative simplicity of the domain. A dependent description depends on descriptions of other cues than the one to which it attaches itself. Although dependent descriptions are already implemented in *The Ear's Mind*, since their working relies on other mechanisms that lie beyond the first implementation phase (which defines the current project), only independent descriptions will be discussed here.

We start by defining the independent-description class:

```
(defclass independent-description (workspace-structure)
  ;;defining the description class
  ((object :initarg :object :accessor :object :writer :set-object)
  ;;the object this independent description is attached to
   (descriptor :initarg :descriptor :accessor :descriptor :writer
    :set-descriptor)
  ;;the descriptor's name. this name is used by all functions which
  ;;work with independent-descriptions.
   ))
```

Class independent-description inherits from the workspace-structure class:

```
(defclass workspace-structure nil
  ;;defining the workspace-structure class.
  ((structure-category . . .)
  ;;what category the structure is
  (proposal-level . . .)
  ;;for internal bookkeeping, one of the integers 1, 2, and 3
  ;;depending on the phases between proposing and building the
 ;;structure. This is needed since instances of proposed structures
 ;;have to be created even if they don't get built at the end.
   (family . . .)
 ;;the family the structure belongs to (see bunches and
 ;;families)
   (tcl-handel . . .)
 ;;the graphics handle of this structure. This numeric handle is
 ;;returned from the graphics server at creation time of the
 ;;structure, and is used for all subsequent graphics activities.
))
```

Now we can create a new independent description instance using:

```
(defun make-independent-description
  (object descriptor &aux new-description)
;;descriptor should be the name of the Slipnet node which describes
;;the object. The object is a cue for now, but in the second phase
;;of The Ear's Mind it could be groups of objects as well (i.e.
;;groups of cues or groups of groups etc.)
  (setq new-description
   (make-instance 'independent-description :object object
              :descriptor descriptor))
;;return the new-description
  new-description)
```

And add an independent description to an existing object (such as a cue):

```
(defun add-independent-description (object descriptor &aux desc)
;;descriptor is the symbol name of the descriptor node in the Slipnet
  (:set-instance-list
   (append (:instance-list descriptor) (list object))descriptor)
  ;; append the object to the instance-list of the node corresponding
  ;;to the independent-description.
  ;;returns the new description, not the list of all descriptions of
  ;;object
  (setq desc (make-independent-description object descriptor))
  (workspace-add-structure desc (:level object))
  (:set-independent-descriptions
    (append (:independent-descriptions object) (list desc)) object )
  ;;return the new description
  desc)
```

Other functions directly built on independent descriptions are (among others):

```
(defun remove-independent-description (object descriptor &aux description)
  ;;descriptor input is the symbol name of the descriptor node
  ;; remove the object from the instance-list of the node
  ;;corresponding to the independent-description
. . .
;;return the object from which the description was removed
object ))

(defun has-ind-desc-p (descriptor object)
  ;;retruns true if at least one of the independent descriptions of
  ;;the given object is the given descriptor
  . . .)
```

7.5.4 The graphics server

The Ear's Mind activities are taking place on the time-frequency input domain, where input cues are defined, descriptions are attached and bonds and are built and destroyed. Codelets 'land' on cues in specific zones, and 'sniff' for other cues in certain directions. All this activity would be very hard to inspect and analyse without an ability to watch it happening. Since the model is based on emergent behaviour, it is of great importance (and joy) to be able to assess the evolving activity throughout the run rather than inspecting the outcome only. This calls for the implementation of a real-time graphics window on which all the desired activity could be drawn. Here, real-time is not used in the real sense of the word. Rather it gives the impression of watching things as they evolve. Since the absolute timing of *The Ear's Mind* in the current implementation is irrelevant to its function, there are no hard real-time demands on the graphics server either. In other words, the graphics server is allowed to slow down *The Ear's Mind* as long as it does not slow it down to a halt. For portability reasons, I chose to implement the graphics server using socket communications so that the server will not only be written in a language that supports graphics, but also be able to run on different operating systems and hardware. I therefore chose tcl/tk, which offers all the above requirements. On the tcl/tk scripting language, see (Welch et al 2003). Below I sketch my implementation of the graphics server. This entails describing some of the implementation details, and listing the capabilities the server offers.

The Ear's Mind runs as compiled Lisp code in a separate process from the graphics server, which is coded in tcl/tk. At initialisation time, the Lisp code creates a socket connection with the graphics server. Needles to say, I have equipped both processes with standard socket communication code and adopted a primitive text protocol to serve for the simple needs of *The Ear's Mind*. The graphics server uses a callback function which processes input from the Lisp engine and returns output. Each time an input line from the Lisp code is available, the graphics server evaluates that line as if it was a tcl/tk input. In this straightforward way, the socket interface acts as a wrapper for tcl code from within the Lisp code. All the basic graphics functions are implemented in the graphics server and called from the Lisp code. Each call contains parameters which are evaluated by the called procedure. The called procedure then sends back a value to the Lisp code. A simple error catch is implemented for signalling problems. The main use for the returned values from tcl to Lisp is the graphics handle. It is assigned by the graphics server to each newly defined graphics object. The graphics server is in charge of keeping track of all the different graphics handles. These handles are used by the Lisp code in all subsequent calls when referring to existing graphics objects.

The Lisp code sends a line to the socket connection using function 'tell-gs'. It writes the given string as a line to the previously defined and opened socket. It then reads a line from the socket and returns it to the calling function.

```
(defun tell-gs (str)
;;send str as a line to the socket sock and return
;;the line read from the socket.
  (write-line str sock)
  (read-from-string(read-line sock)))
```

The following function is used for easily defining new graphic calls. The parameters used when calling this function will define the function for all subsequent uses. In all subsequent code, 'gs' refers to 'graphics server'.

```
(defun make-gs-command (command)
  #'(lambda (&rest arg-string)
           (tell-gs (string-append command " " (args2str arg-string)))))
;make a new function which will append command (string) to the string
;of arg-string and call tell-gs on it. tell-gs will be compiled in-
;line, and returns the values read from read-line converted from the
;read string of the socket.
```

Once make-gs-command has been defined, it can be used in the following way:
Define a new function 'draw-poly' with make-gs-command

```
(setq draw-poly (make-gs-command "draw-poly"))
```

Now draw-poly is a symbol which can be evaluated by funcall, so calling

```
(funcall draw-poly "2 4 6 blue")
```

Will send the line 'draw-poly 2 4 6 blue' to the graphics server which will then call the tcl procedure draw-poly with the parameters 2, 4, 6, and 'blue'.

```
(setq gs-draw-poly . . .)
   ;;draw a polygon with the given coordinates and colour.
   ;;returns id handle

(setq gs-draw-rect . . .)
   ;;draw a rectangle line with the given coord lt ht lf hf.
   ;;returns id handle

(setq gs-delete . . .)
   ;;delete the object with id id. returns id handle

(setq gs-delete-after . . .)
   ;;delete object with id handle after delay time in milliseconds"

(setq gs-config . . .)
   ;;change the graphics attribute of object with id to value
   ;;id attribute value

(setq gs-start-blink  . . .)
   ;;start blinking object with id time color
   ;;with time milliseconds intervals. Use function gs-stop-blink
   ;;to stop blinking and reverting to original color

(setq gs-draw-grid . . .)
   ;;draws a grid of horizontal lines with a given interval freq
   ;;difference and colour
```

Some of the functions implemented this way are:

```
(defun draw-cue (cue)
  ;;if the cue is already on screen quit, otherwise, create a
  ;;graphics object for the cue, draw it, and return its handle.
  ;;Shape and colour of drawn cue depend on the existence and type of
  ;;the attached independent description
  (if* (:tcl-handel cue) then ()
       ;;if cue has already a graphics handle, return
     else
       (princ ".")
        ;;print dot to the terminal (indicating progress when many
        ;;cues are printed in one go
       (if* (and (eq 1 (:level cue)) (:independent-descriptions cue))
       then
       (draw-cue-level1-desc cue)
        ;;if the cue is of level 1 and has an independent description
       ;;then draw it according to its description type
       else
       (setq maxt (:max-time cue) maxf (:max-freq cue)
             mint (:min-time cue) minf (:min-freq cue))
       (funcall gs-draw-poly mint maxf maxt maxf maxt minf mint minf
              "LightBlue")
       ;;Draw a regular light blue rectangle for the cue)))


(defun draw-cues (list-cues)
  ;;draw all the cues from the given list of cue instances
  ;;calls the above function for each cue
  . . .)
```

Using low-level functions such as the above examples, functions were implemented which enable higher-levels of interaction with the graphics server:

```
(defun config-cue (cue attribute value)
  "send tcl the command to change the attribute of cue to value"
  (funcall gs-config  (:tcl-handel cue) attribute value))

(defun delete-cue (cue)
  ;;deletes a cue from gs
. . .)

(defun draw-zone (lt ht lf hf sniffing-cue)
  ;;draw a rectangular zone around the sniffing-cue, for details
  ;;see zones
  . . .)

(defun plotlog ()
;;set logarithmic frequency plotting mode and redraw the screen
  (funcall gs-plot-log)
  (clear-screen)
  (funcall gs-draw-grid 100 'gray)
  (draw-cues (all-workspace-objects)))

(defun plotlin ()
;;sets linear frequency plotting mode and redraws the screen
  . . .)
```

On the tcl side of the socket, things go as follows. After setting up a scrolled canvas for the display (see Welch et al 2003), the graphics server initialises the socket connection. Currently, the display is set to an area of 1000 by 600 pixels. All communications with the graphics server were implemented for complete abstraction of display details. Consequently, and dimensions are given in seconds and Hertz in the Lisp code. The graphics server converts these dimensions to an internal display representation. In addition, zoom functionality has been implemented to allow for zooming in and out and inspecting activity in details. All this activity is done entirely without the knowledge of the Lisp engine. Some other implemented functions are mentioned below:

- Functions for the numeric calculation of geometries involved in the plotting of the lines, splines and triangles used by the graphics server.
- Functions facilitating the user interface such as some mouse and key bindings together with rudimentary zooming and dragging capabilities. Both zoom and translation functions may be used during a run. In such a case, already drawn structures zoom and translate, while newly drawn ones are plotted in the current zoom factor and correct position relative to the original domain.
- Functions were defined which tag cues, currently built bonds, and destroyed bonds separately and make it possible to toggle their display on and off. This mode greatly helps viewing the system's behaviour during and after the run.
- Procedures for saving postscript images at any stage from the Lisp engine.

7.5.5 The Slipnet

Copycat's Slipnet is tailored for the specific domain of letter string analogies. Since Copycat learns nothing between runs, the domain specific knowledge it has resides in the Slipnet and in the definition of Codelets. When these function together, we have active concepts in the Slipnet steering the activity of Codelets. At the same time, bottom-up Codelets, which are not sent by nodes in the Slipnet are launched, and compete for priority on the Coderack with top-down Codelets. The Slipnet defines the (dynamic) relations between nodes in the domain at hand. It defines the links for spreading activation between nodes and handles the variable length some links have. In addition, the Slipnet registers which Codelets should be launched by which nodes. The Codelets, on the other hand, know how to choose a number of letters and consult the Slipnet for relations between the letters. Codelet action is split in Copycat into three phases. First, a structure has to be proposed. If it happens to be proposed, a tester Codelet is put on the Coderack with a given priority for later execution. If the tester Codelet is launched, it tests the proposed structure and if it turns out to be strong enough (taking into account the state of other letters in the area) then a builder Codelet will be posted onto the Coderack. Finally, when the builder is launched, it will check for competing structures and fight with them if necessary. If it wins at the end, the proposed structures (e.g. a bond between two letters) will be built.

From the above sketch (for more details see chapter 4) it should be clear that the Slipnet forms the only global static definition (although its behaviour is dynamic at run time, the defined knowledge is static), holding together the entire domain. Consequently, to get things started an initial Slipnet structure has to fit the auditory domain. Since any Slipnet structure of such a domain can only be verified through experimentation, the initial Slipnet should only be regarded as a sketch to get things going. However, technically, the mechanics of defining the Slipnet in Copycat were not as straightforward as one would wish. Therefore, I have implemented functions to simplify the definition and initialisation of the Slipnet.

The `make-Slipnet-obj` function is defined for the production of Slipnet nodes and links including all their parameters. The idea is to have one central point from which all Slipnet

definition of instances could be made. Additionally, information only has to be given regarding the connectivity of each node. At initialisation time, a Slipnet-building function scans all the given definitions, creates their instances and interconnects them. In that way, no specific order has to be followed in the definition code, and code redundancy is reduced. See below for an example of the definition of section of the initial Slipnet for *The Ear's Mind*.

```
;;define four cue nodes: plato-positive-temporal-derivative,
;;plato-negative-temporal-derivative,plato-positive-spectral-
;;derivative, and plato-negative-spectral-derivative.
;;for each attach an outgoing link with fixed length 82 of type
;;'category' leading to the node 'derivative'. Node 'derivative'
;;was not yet defined. However, after all definitions have been
;;given, a scanning function takes care of matching all the
;;separate pieces.
;;
(make-Slipnet-obj :nodes '((plato-positive-temporal-derivative
                plato-negative-temporal-derivative
                plato-positive-spectral-derivative
                plato-negative-spectral-derivative))
          :out-links '((:link-type 'category :to-node 'derivative
                     :fixed-length 82)))

;;form outgoing links of type 'category' leading to node 'positive'
;;from nodes plato-positive-temporal-derivative and plato-positive-
;;spectral-derivative. since the two nodes were already defined
;;above, the scanning  ;;function will not make new instances
;;thereof, but only define their new connectivity
;;
  (make-Slipnet-obj :nodes '((plato-positive-temporal-derivative
                plato-positive-spectral-derivative))
          :out-links '((:link-type 'category :to-node 'positive)))
;;
;;form outgoing links of type 'category' leading to node 'negative'
;;
 (make-Slipnet-obj :nodes '((plato-negative-temporal-derivative
                plato-negative-spectral-derivative))
          :out-links '((:link-type 'category :to-node 'negative)))

                                  .
                                  .
                                  .

;;now define the category nodes


(make-Slipnet-obj :nodes '((derivative negative positive spectral
temporal)))
```

Additionally, Codelets can be declared in the same way as the above, together with any other types of links and nodes. New keywords can be easily implemented for adding novel structure elements to the Slipnet. See figure 7-7 for a graphic representation of the definitions above following Slipnet initialisation.

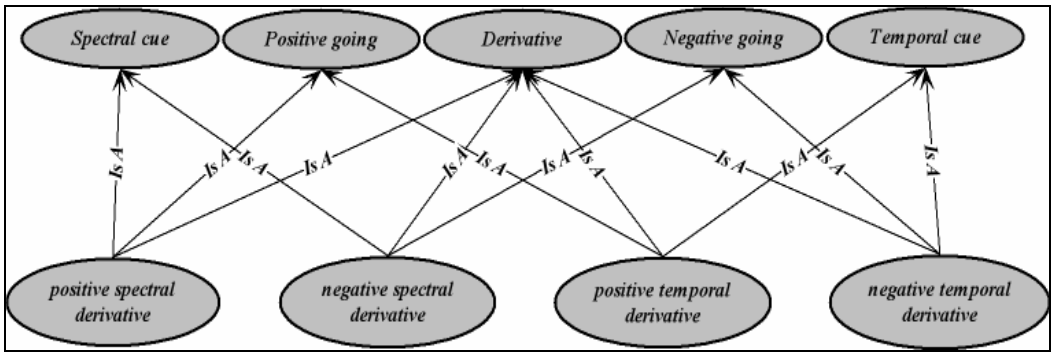

Figure 7-7: defining spectral and temporal derivative cues for The Ear's Mind's Slipnet

Figure 7-8 below depicts the original Copycat Slipnet for comparison with the initial Slipnet definition of The Ear's Mind (given in figure 7-9).

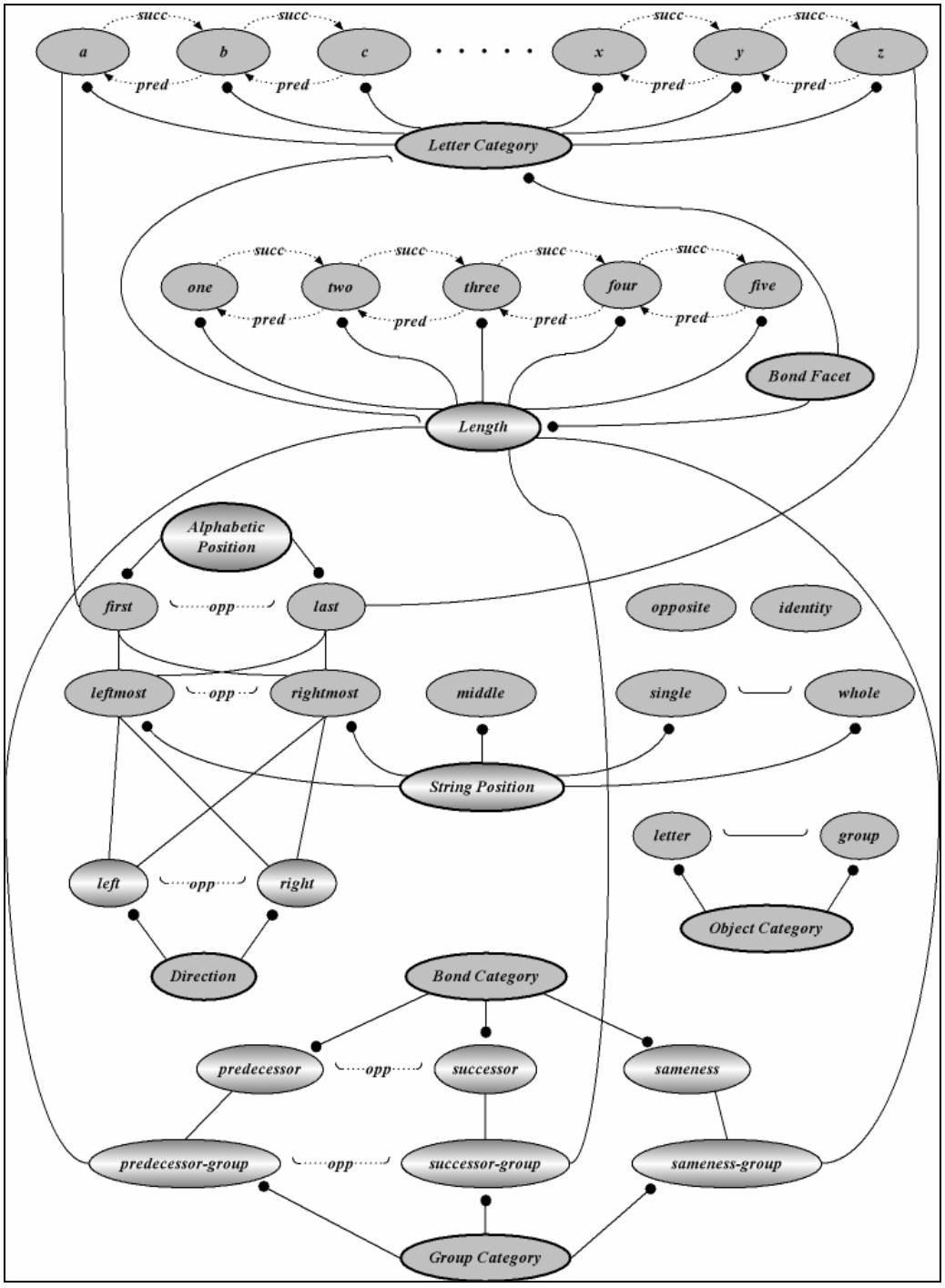

Figure 7-8: Copycat's Slipnet

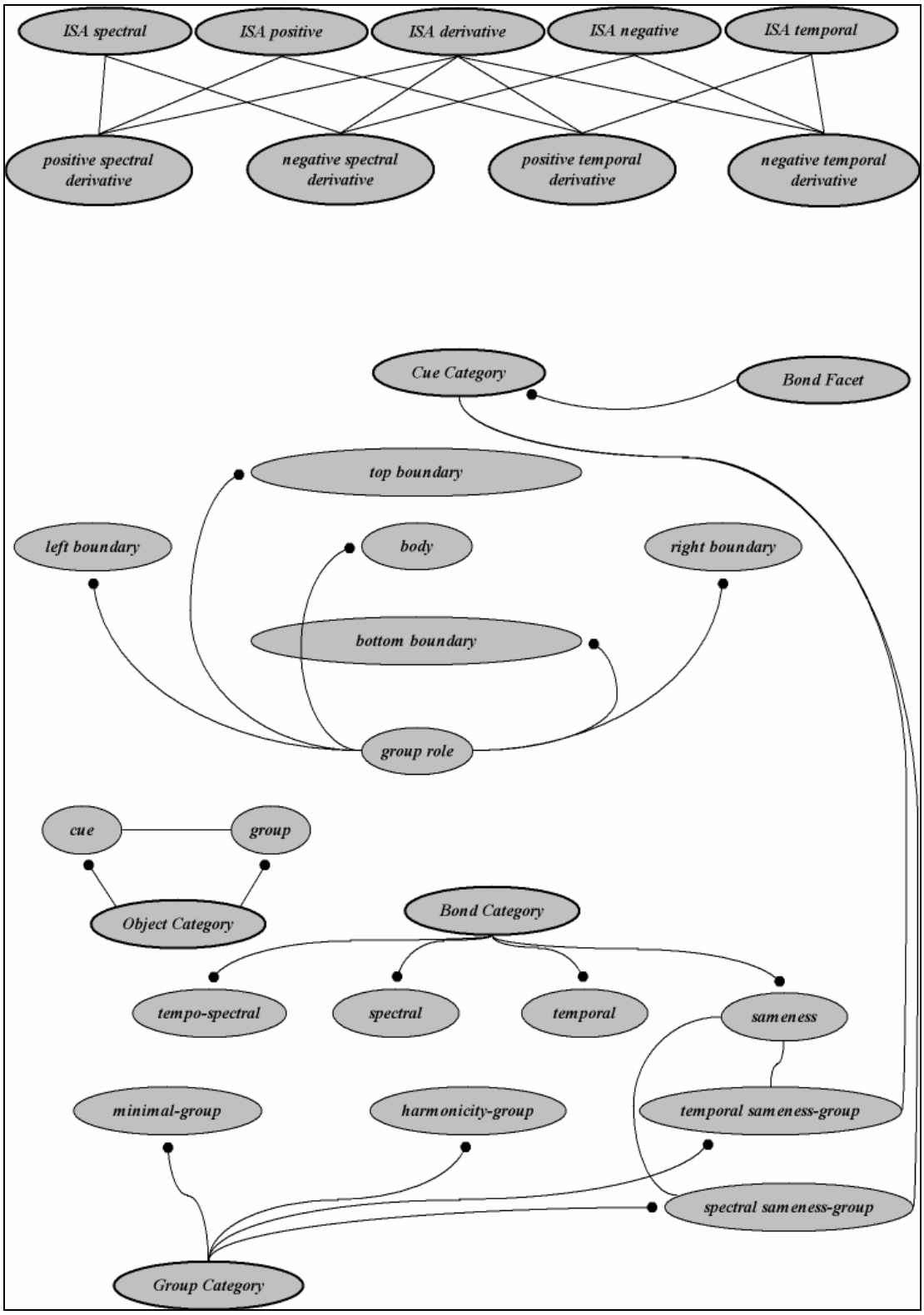

Figure 7-9: The Ear's Mind's Preliminary Slipnet

7.5.6 Zones

One of the major differences between Copycat's original domain and the auditory domain of *The Ear's Mind* is the way in which input elements are organised. Since these element form the substrate on which all activity takes place, it is of great importance to utilise their implicit organisation in the model's functionality. In Copycat, this has been done to such extent, that Copycat can be truly said to be hard-wired to work in the letter-string domain. Copycat was programmed to follow the static global input structure of its domain. It 'knows' of the existence of three strings of letters, which it can access independently. Within each string, it can access each individual letter and its neighbours, and find out whether it is the left-most or right-most letter. Needles to say, this has nothing directly to do with Copycat's abstract architecture – it is more of a shell surrounding the internal working of the model, and interfacing Copycat with three strings of letters. Now, *The Ear's Mind* works in an entirely different environment, where no prior knowledge exists about the internal organisation of input elements. Every element might have a different size, and occupy a different location. Moreover, there is no neat division among three strings of elements. Actually, no subdivision exist whatsoever. The following example should help clarify the problem:

In Copycat, a scout may need to relate adjacent letters within the same string. Since the letters are prearranged into strings (by the very definition of the domain), it is simple for the scout to choose a string, and proceed with randomly choosing a letter within that string. Once done, it can choose one of the letters to the right or left (if both exist). In *The Ear's Mind*, such neatly organised input is not available. Here, the input elements are spread throughout a two-dimensional surface. How should a scout go about choosing two or more cues? Obviously, we must first know the type of relationship the scout is investigating. It could be searching for a pair of adjacent cues to suggest that they should be regarded as each other's extension. Such might be the case when two adjacent first derivative cues are of the same type and direction and their relative positions suggest that a larger cue be made by union (see Figure 7-10). In this case, obviously, the two cues must be adjacent.

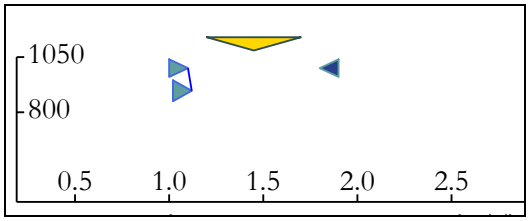

Figure 7-10: Two adjacent first derivative cues of the same type and direction. Their relative positions suggest that a larger cue be made by union.

A different situation would arise, for instance, when relating cues by proximity (see Figure 7-11 below). The idea here is to choose a cue and then to look in the area around that cue for other cues of the same type. A bond might be built between two cues that are closer to each other than to other cues in the vicinity. Here, however, demanding the cues to be adjacent (i.e. with no other cues occupying the space in between) seems to be over constraining. In real-world input, the existence of small non-relevant cues spread throughout the domain is nearly guaranteed. Such cues would completely block the proximity principle. It seems clear, then, that we must let every scout manage its own choice of elements. The relation the scout is looking for among elements should constrain its choice of elements. This explains the need for a reimplementation of the Codelet interfacing mechanism in *The Ear's Mind*. Since in Copycat only two cases occur (either choosing two adjacent letters within one string or choosing two letters in corresponding positions

in different strings), Codelets first *choose elements,* and only then, by inspecting their properties *choose a relation to use.* In *The Ear's Mind*, I have found it necessary first to *choose a relation*, and only then, when knowing what *type* of element is needed, *choose the elements.* Consequently, in *The Ear's Mind* there exist far more bottom-up Codelets then in Copycat. Each of *The Ear's Mind*'s Codelet comes with its own taste of element choice.

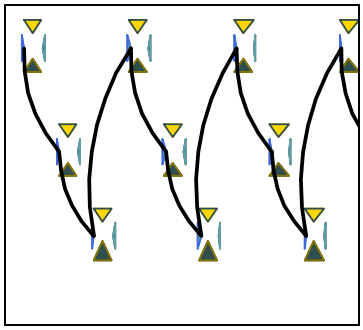

Figure 7-11: resulting stream of The Ear's Mind's proximity scouts.

By analogies with self-organising biological systems (see Camazine 2001), I have devised a mechanism for choosing elements while keeping things as local as possible (both in terms of time-frequency space and computational sharing of global knowledge). The following points summarise the requirements for the Codelets – time-frequency domain interface.

- To keep the search for elements local, use search zones. Only the elements in a given zone can be seen.
- The above enables the future integration of the zones themselves in directing *The Ear's Mind*'s activities. In this way, zones could have relevance and direct more activity to specific parts of the time-frequency domain.
- Zones should be implemented which are built on a given element. Since the behaviour of such zones is analogous with sniffing the air in a certain direction for scent evidence of pray, I name such zones 'sniffing zones'
- After 'landing' on an element (e.g. a cue), a sniffing zone should be built starting from a given tf-point of that element, and defining an area according to the given directional and dimensional parameters.
- When a so-called 'sniffing cue' uses the sniffing zone, it should be able to sniff for elements of specific characteristics according to its own taste (e.g. absolute of relative size, connectedness, age, type of cue, etc.)
- To allow results from different zones to be compared (e.g. for resolving fights), the zone system should be based on a universal scoring system.
- Such a system should relate all relevant parameters relative to the dimensions of the zones themselves, so that comparison of parameters from different zones would still be possible.

For implementing the above, and still keep computational complexity reasonable, I have opted to rectangular zones. If future investigations will show that other zone shapes are critical to the working of *The Ear's Mind*, then changes can be easily made. Implementation details of the more important sections of the zone definition file follow below.

To simplify zone definition (since many zones might be tested and discarded at the experimentation phases, when the zone strategies are investigated), I have defined some Lisp macros which produce the zone definition code automatically.

The Lisp macro make-zone creates a closure at compile time containing the code for the definition of a new function.

```
(defmacro make-zone (from-where list? &body body)
;;all zones return the multiple values: start-time end-time start-
;;freq end-freq

;;if parameter from-where is the symbol 'list', then a list of (time
;;freq) is given at run time and the right code should be used at
;;compile time for binding the coordinates to the code. The macro is
;;used with defun to define a function with make-zone to create a
;;closure at compile time with the right body of code. If the input
;;is not equal to 'list', then  the body should access the
;;coordinates through object slots.
;;look at the input parameter at function definition time:
  (let ((from (gensym)))
    (if list?
     ;;if it's 'list' then bind the time and frequency (given as
     ;;parameters to the defined function at run time as input
     ;;parameters to the code body.
   `(let ((,from ,from-where))
     (with-gensyms (time freq)
       (let ((time (first ,from)) (freq (second ,from)))
             (values ,@body))))
     ;;otherwise, extract the time and freq coordinates from the
     ;;object given at run time and bind them to the code defined
     ;;at compile time.
   `(with-gensyms (min-time max-time min-freq max-freq )
     (let ((from ,from-where) (min-time (:min-time ,from-where)) (max-time
(:max-time ,from-where))
     (min-freq (:min-freq ,from-where)) (max-freq (:max-freq ,from-where)))
       (values ,@body))))))
```

New zone functions can now be easily defined by using the make-zone macro in the following way:

we define a landing zone function which is given values for time (in seconds) and frequency (in Hertz) coordinates and returns the four coordinates for the rectangular zone: minimum frequency (= given frequency value) and maximum frequency (=given frequency value * 3), and minimum time (=given time value) and maximum time (= given time value + 1.5 seconds) values.

```
(defun landing-zone (l &key ) (make-zone l 'list time (+ time 1.5)
freq (* freq 3)))
```

Using the same macro, we now define a time-to-time zone. This zone is used when a Codelet lands on a (positive) temporal derivative cue and sniffs in the positive time direction for (negative) temporal derivatives. Here, time and frequency values are taken from the cue (given at run time), so the parameters min-time, max-time, min-freq, and max-freq are already bound to the right values taken from the cue slots and can be used in the code body. Calling the t-t-zone at run time will bind the corresponding coordinates of the given cue to the code and return minimum frequency (cue min-frequency – 200 Hz.), maximum frequency (cue max-freq + 200 Hz.), minimum time (cue max-time), and maximum time (cue max-time + 1 sec.). Following definition, the function can simply be called by with (t-t-zone cue). From the definition it can be seen, that the zone will stretch in time from the temporal end-point of the sniffing cue to 1

second past that point in the positive time direction, and have a frequency range covering the entire cue padded with 200 Hertz on each side.

```
(defun t-t-zone (cue) (make-zone cue nil max-time (+ max-time 1) (- min-
freq 200) (+ max-freq 200)))
;;the above original was (+ max-time 1)
```

Likewise, other zones were defined for different uses. Some example zone definitions follow:

- a time-up-frequency zone (for sniffing from a temporal cue to a frequency cue in the upper frequency range):

```
(defun t-f-up-zone (cue) (make-zone cue nil min-time (+ min-time 0.2)
      (- max-freq 50) (+ max-freq 50)))
```

- a time-down-frequency zone (for sniffing from a temporal cue to a frequency cue in the lower frequency range):

```
(defun t-f-down-zone (cue) (make-zone cue nil min-time
     (+ min-time 0.2) (- min-freq 50) (+ min-freq 50)))
```

- a frequency-frequency zone (for sniffing from one spectral cue to another

```
(defun f-f-zone (cue) (make-zone cue nil
       (- (middle-point max-time min-time) 0.5)
       (+ (middle-point max-time min-time) 0.5)
                  min-freq (* min-freq 2.2)))
```

Notice that some zones compute their frequency coordinates as ratios (rather than differences) of frequencies. This is due to the logarithmic nature of auditory perception and natural harmonics and allows capturing harmonically related ranges symmetrically below and above a given frequency.

Other defined zone types are freq-time-up zone, freq-time-down-zone, frequency-extension-zone, time-extension-zone, time-proximity-zone and freq-proximity-zone. Needles to say, not all zones prove effective for *The Ear's Mind*'s activity. Since defining new zones is very easy with the above macro, different zones could (and will) be tested. Since the above functions return only zone coordinates, we need some functions to create the zone itself, and actually use it.

The following function returns the instances of all objects whose time-frequency vertex of interest lies within the given zone coordinates. Optionally, only objects are returned, whose attached description matches the given description type. An example use of this function is when searching for all positive temporal derivative cues whose lower left vertex (minimum time and frequency values) lie within the range of the given zone coordinates.

```
(defun find-cues-in-zone (start-time end-time start-freq end-freq point-
cat &optional desc &aux instances)
  ;;returns all the instances which reside within the given time and
  ;;freq range, whose tf-points are of point-cat and whose desc is
  ;;desc
  ;;first, find all tf-points in the given range.
  (setq instances (mapcar #':cue (intersection
      (find-tf-points start-time end-time 'time-vector point-cat)
      (find-tf-points start-freq end-freq 'freq-vector point-cat))))
  ;;now, if no instances found, return nil and quit. Otherwise:
  (if instances
     ;;if a description was given, then remove all instances which
     ;;do not have the given description attached to them.
      (if desc (remove-if-not #'(lambda (obj)
         (has-ind-desc-p desc obj)) instances)
  ;;finally, return the instances found (if none found returns nil)
  instances)))
```

Next, some sniffing functions can be implemented:

To be able to define new sniff functions with minimum coding, I have defined a uniform set of bound parameters to be used by all sniffing functions. The function may or may not use them, but it can rely on their existence if needed. The defined parameters are:

- zt-span – is the temporal span of the sniffing zone, i.e. maximum time value – minimum time value in seconds.
- zf-span – is the spectral span of the sniffing zone, i.e. maximum frequency value – minimum frequency value in Hertz.
- ft-span – is the from time value denoting the centre temporal position of the sniffing instance in seconds
- ff-span – is the from freq value denoting the centre spectral position of the sniffing instance in Hertz.
- delta-t – is the slot delta-t of the sniffing instance denoting the maximum time value – minimum time value in Hertz.
- delta-f – is the slot delta-f of the sniffing instance denoting the maximum freq value – minimum freq value in Hertz.

Without getting into too much detail, sniffing functions can be easily defined in the following ways:

1. The macro sum-sniff-params can be used to combine the results of (normalised) scores. In case of normalised input scores, it returns a score between 0 and 100 which can be used for comparison between zones of different sizes.
2. basic macros can be defined for computing distance norms such as:

```
(defmacro freq-span  ()      `(min 1 (/ delta-f zf-span)))

[insert math functions for these computations]

(defmacro freq-span2 ()      `(min 1 (expt (/ delta-f zf-span) 2)))

(defmacro freq-prox  () `(expt (* 0.22 (log (/ center-f ff) 2)) 2))

(defmacro time-prox  () `(expt ( - center-t ft) 2))
```

3. functions can then be defined using the above macros:

```
(setq size (sum-sniff-params  (freq-span) (time-span)))

(setq distance (sum-sniff-params (freq-dist) (time-dist)))

(setq big-and-close (sum-sniff-params
                     (freq-dist)(time-dist)(freq-span)(time-span)))

(setq proximity-measure (sum-sniff-params (freq-prox) (time-prox)))
```

4. The macro 'sniff' uses the function symbols (e.g. size, distance or big-and-close) such as the ones defined above. It then applies the returned lambda expression to the given list of cues. The macro produces a function which sniffs each cue from the list of cues using the given sniffing method. This function returns a list containing a sniffing score for each cue.

```
(defmacro sniff (fun ilst (ist iet isf ief ift iff))
  (with-gensyms  (st et sf ef ft ff lst)
    `(let ((st ,ist) (et ,iet) (sf ,isf) (ef ,ief) (ft ,ift)
           (ff ,iff) (lst ,ilst))
      ;;after binding external parameters to internal ones
      ;;for the safe execution of the macro
      ;;
      ;;expand to code which will call the given function fun.
      ;;in the definition below all the parameters that need
      ;;to be defined and bound for every function of this type by
      ;;the interface definition are computed and given to
      ;;the function. The result is coerced into float format
      ;;and added to a list. The function is called on each cue
      ;;from the given list of cues ilst. The list of scores is
      ;;finally returned.
      (mapcar #'(lambda (sniffed)
                  (coerce
                    (funcall ,fun (- et st) (- ef sf) ft ff
                         (:delta-t sniffed) (:delta-f sniffed)
                         (:time-stamp (:center sniffed))
                       (:freq-stamp (:center sniffed)))
                    'float))
                lst))))
```

As an example use of this macro, I have included below a line from the 'make-scout' macro. In this example, the starting and ending values of time and frequency to be used by the sniffing method are bound by the make-scout macro. The sniffing method to be called is bound to symbol from-method. These bindings are done at compile time. The make-scout macro evaluates to code at compile time that will call the sniff function at run time with the correct parameters. This results in the list of scores based on the given sniffing method to be stores in score-list.

```
(setq score-list (sniff ,from-method cue-list (st et sf ef 0 0 )))
```

In some cases, the sniffing will be used in comparing scores from different zones at different locations with different dimensions. The idea is to generate all scores normalised between a top 100 and a bottom 0 (if scores are to be summed) or a bottom 1 (if a score product will be computed). This makes it easy to combine scores from different parameters, and makes the scoring system 'true'. Moreover, scores may be combined together to form compound scores. To achieve this, each parameter should be relative to a local measure. That local measure was chosen to be the relevant measurement of the zone itself. This results in measurements that are all relative to their zones. This outcome is desired since we strive to perform all operations locally (e.g. with no global computations and no measuring functions that scan the entire domain). At the same time, the normalisation still allows us to compare and relate measurements among different zones.

7.5.7 Scouts and Bonds

In the previous section, I have discussed the differences in the Codelet interfacing mechanisms between Copycat and *The Ear's Mind*. For the Codelets code itself, implementation consequences for were twofold. First, since the implementation of each Codelet is more self-contained (i.e. less distributed in generic chunks of code) it is far easier to implement new types of Codelets or to alter their behaviour. This makes for easy experimentation (when many proposed Codelets may be tested only to be discarded when proved useless). Secondly, defining multiple Codelets, each with self-contained code, results in much more source-code than seems necessary. This makes the code difficult to read and maintain. To solve this problem I have implemented Lisp macros that handle much of the definition of Codelets automatically. In other words, these macros produce source code which in turn implements each Codelet.

Every scout works in the same fashion. The make-scout macro is used to produce code at compile time that results in a 'hard-wired' function containing all the internal functions used by the scout. A scout in *The Ear's Mind* executes the following steps:

1. Land on a random spot. In the current implementation, this spot is uniformly distributed (corrected for the logarithmic nature of the frequency dimension). Future expansions will introduce relevance related Slipnet control (based on successful, exhausted, futile zones etc.)
2. Create a landing zone from this spot. Its dimensions compensate for the logarithmic nature of the frequency dimension.
3. Return a list of all the cues in the landing zone, which conform to a desired tf-point slot and an independent description type.
4. Based on a desired sniffing method, sniff the cue list, and score each cue, producing a list of scores.
5. Choose a winner and call it the sniffing cue.
6. Based on a desired sniffing zone type, create a sniffing zone extending from the sniffing cue. The desired sniffing zone function determines zone dimensions, direction, location etc.

7. Return a list of all the cues in the sniffing zone, which conform to a desired tf-point slot and an independent description type.
8. Based on a desired sniffing method, and a sniffing cue tf point, sniff the list of cues score each one, returning a list of scores.
9. Based on a desired choice method, choose a winner cue using the list of scores. Call this winner the sniffed cue.
10. Propose a bond between the sniffing and the sniffed cue.

The execution steps listed above and the variety of choices unique for each predefined scout, make scout definition an ideal case for defining a macro that automatically produces the source code of each scout. Using such a macro makes the code more readable, and at the same time enables easy and quick definition of new types of scouts. With the new make-scout macro, we can define a new scout with very little code. The choice of the initial sniffing cue is done through the use of a landing zone in order to ensure that while larger (more salient) cues are probabilistically preferred (i.e. there is always some chance of smaller cues to be chosen), the choice is made locally within a given landing zone. Without such a zone, cues will be chosen globally, and fail desperately.

Before describing the macro itself, here is an example showing the definition of a new scout. The scout will be called pos-temp-to-neg-temp-scout, denoting a scout which lands on positive temporal derivative cues and sniffs for negative temporal derivatives.

The scout is defined by calling macro make-scout to produce code with the following attributes:

Family type of proposed bond:	**internal bonding family**
Land on cues whose:	**centre is in the landing zone**
And have the description:	**positive temporal derivative**
Stochastically prefer:	**larger cues**
From sniffing cue build a:	**temporal to temporal sniffing zone**
Find cues whose:	**centre is in the sniffing zone**
And have the description:	**negative temporal derivative**
Stochastically prefer:	**large cues close to sniffing cue**
Sniff from:	**centre of sniffing cue**
Pick a winner by preferring:	**probabilistically higher scores**

Here is the code for implementing this scout:

```
(setq pos-temp-to-neg-temp-scout
   (make-scout
      internal-bonding-family
      'pos-temp-to-neg-temp-scout
      ('center "PLATO-POSITIVE-TEMPORAL-DERIVATIVE" size t-t-zone)
      ('center "PLATO-NEGATIVE-TEMPORAL-DERIVATIVE" big-and-close2
       'prob :center)))
```

Following definition, the scout can be called by evaluating:

(funcall pos-temp-to-neg-temp-scout)

The make-scout macro is implemented in the following way:

```
(defmacro make-scout
   (family bond-category
     (from-tf-points from-descriptions from-sniff-method
      build-zone-type)
     (to-tf-points to-descriptions to-sniff-method choice-method
      sniff-from-tf-point))
;;
;; make a new scout. returns a lambda closure which can be called by
;; a funcall following a setq.
;;
;; arguments:
;;    family is the family to which the proposed bond will belong
;;       (see families and bunches)
;;    bond-category is used by later bond testing and building
;;       functions
;;    from- and to- refer to args belonging either to the from-obj
;;       (the sniffing obj) or to the to-obj, (the sniffed object).
;;    zone type is the function to be called which returns a zone
;;       (e.g. ;;'t-t-zone' or 'landing-zone').
;; tf-point is the part of the obj to be used when sniffing and
;;       measuring distances (e.g. center, top-right).
;;    description is used to choose only objects of that description
;;       (e.g. "PLATO-POSITIVE-TEMPORAL-DERIVATIVE")
;;    choice-method (one of 'highest, 'lowest or 'prob) indicates the
;;       use made of sniffing list of scores: 'highest and 'lowest
;;    probabilistically favour the highest or lowest scores
;;       correspondingly while 'prob chooses uniformly from the list.
;;    sniff-method is the method to be used for sniffing (e.g. size,
;;       big-;;and-close2).
;;    family refers to the family class (not the bond-family class)
;;       which ;;groups some attributes per family.
;;
;; first, bind all arguments to internal ones
;;
;;
 (with-gensyms (fam bond-cat from-tf from-desc from-method zone-type
                to-tf to-desc to-method choose sniff-from-tf)
  (let ((fam family) (bond-cat bond-category)
        (from-tf from-tf-points) (from-desc from-descriptions)
        (from-method from-sniff-method) (zone-type build-zone-type)
        (to-tf to-tf-points) (to-desc to-descriptions)
        (to-method to-sniff-method) (choice choice-method)
        (sniff-from-tf sniff-from-tf-point))
```

```
;;
;; Return a lambda expression containing the following steps:
;;
   `(lambda
      (&aux st et sf ef cue-list score-list from-obj
        to-obj bond-facet)
;; 1. Stochastically land somewhere.
;; 2. Set st, et, sf, ef to zone coordinates according to the given
;;     landing-zone.
   (multiple-value-bind (st et sf ef) (landing-zone (land-somewhere))
;; 3. Ask the graphics server to draw the new zone for 600
;;    milliseconds
   (draw-temp 600 gs-draw-rect st et sf ef)
;; 4. Create a list of cues containing the given type of cues
;;    (according to their attached descriptions), of which the right
;;    tf point resides within the zone
   (setq cue-list (find-cues-in-zone st et sf ef
                    ,from-tf ,from-desc))
;; 5. If at least one cue was found, then score the cues found by
;;     sniffing them with the given sniffing function (e.g. size)
   (if* cue-list then
    (setq score-list (sniff ,from-method cue-list
                       (st et sf ef 0 0 )))
;; 6. Now, stochastically find a winner based on the scores, and call
;;    it 'from-obj
   (if* (setq from-obj (select-item-by-score cue-list score-list ))
      then
    (format t "~%chose from-obj: ~a~%" (:pname from-obj))
      else (setq from-obj nil)))
;; 7. If from-obj is defined, then create a zone from it of the
;;    right type, direction, size etc. (by calling the given
;;     zone-type function) and then
;; 8. Tell the graphics server to draw the new zone (for 1 second).
   (if* from-obj then
     (multiple-value-bind (st et sf ef) (,zone-type from-obj)
       (draw-temp 1000 'draw-zone st et sf ef from-obj)
;; 9. Now look for the right cues in the zone which match the
;;    given to-tf point and to-description. Save these cues in
;;    a list called cue-list.
       (setq cue-list (find-cues-in-zone st et sf ef
                        ,to-tf ,to-desc))
;; 10. If any cues matched the above search, then sniff
;;     them using the sniffing function ‘to-method’. Save
;;     the resulting list of scores in score-list.
   (if* cue-list then
    (setq score-list
     (sniff ,to-method cue-list (st et sf ef
                (:time-stamp (,sniff-from-tf from-obj))
                (:freq-stamp (,sniff-from-tf from-obj)))))
;; 11. Choose a winner cue using the score-list
;;     and the given choice-method. Call the winning cue 'to-obj.
    (setq to-obj
     (case ,choice
      ('prob (select-item-by-score cue-list score-list))
      ('lowest  (nth (list-min-position score-list) cue-list))
      ('highest (nth (list-max-position score-list) cue-list))))
;; 12. Finally, if the to-obj exists, propose a bond between
;;     the chosen sniffing cue and the chosen sniffed cue.
    (if* to-obj then
      (propose-bond from-obj to-obj ,bond-cat
      ,from-desc ,to-desc ,fam)))))))))))
```

Here is a list of some other scouts defined for *The Ear's Mind* in the same way as the above example:

- *Name* **positive spectral to negative spectral scout**
  Family type of proposed bond: **internal bonding family**
  Land on cues whose: **centre is in the landing zone**
  And have the description: **positive spectral derivative**
  Stochastically prefer: **larger cues**
  From sniffing cue build a: **spectral to spectral sniffing zone**
  Find cues whose: **centre is in the sniffing zone**
  And have the description: **negative spectral derivative**
  Stochastically prefer: **large cues close to sniffing cue**
  Sniff from: **centre of sniffing cue**
  Pick a winner by preferring: **probabilistically higher scores**

- *Name* **positive temporal to positive spectral corner scout**
  Family type of proposed bond: **edge-bonding family**
  Land on cues whose: **bottom left is in the landing zone**
  And have the description: **positive temporal derivative**
  Stochastically prefer: **larger cues**
  From sniffing cue build a: **temp. downwards to spec. sniffing zone**
  Find cues whose: **bottom left is in the sniffing zone**
  And have the description: **positive spectral derivative**
  Stochastically prefer: **large cues close to sniffing cue**
  Sniff from: **bottom left of sniffing cue**
  Pick a winner by preferring: **probabilistically higher scores**

- *Name* **positive temporal extension scout**
  Family type of proposed bond: **edge-bonding family**
  Land on cues whose: **top left is in the landing zone**
  And have the description: **positive temporal derivative**
  Stochastically prefer: **larger cues**
  From sniffing cue build a: **temporal extension sniffing zone**
  Find cues whose: **bottom left is in the sniffing zone**
  And have the description: **positive temporal derivative**
  Stochastically prefer: **large cues close to sniffing cue**
  Sniff from: **bottom left of sniffing cue**
  Pick a winner by preferring: **probabilistically higher scores**

- *Name* **positive temporal proximity scout**
  Family type of proposed bond: **tone-bonding family**
  Land on cues whose: **bottom left is in the landing zone**
  And have the description: **positive temporal derivative**
  Stochastically prefer: **larger cues**
  From sniffing cue build a: **temporal proximity sniffing zone**
  Find cues whose: **bottom left is in the sniffing zone**
  And have the description: **positive temporal derivative**
  Stochastically prefer: **proximity measure scoring**
  Sniff from: **bottom left of sniffing cue**
  Pick a winner by choosing: **the lowest score**

7.5.8 Families, Bunches, and Mutual Support

In previous discussions regarding the attachment of descriptions, and role interpretation plays during the process of perception, I have repeatedly returned to the need of using context as the most important source of information rather than isolated pieces of evidence. Copycat's architecture was designed to make use of existing context through its emergent behaviour, a property that is maintained in *The Ear's Mind* by following the same principles. However, in *The Ear's Mind*, I have found a need for an additional, more direct sort of context, the implementation of which is the subject of this section.

To start things going, we define a 'family' class with a name and colour slots. Now family instances, such as the following definitions, can be defined:

```
(setq internal-bonding-family
   (make-instance 'family :family-name 'internal-bonding-family
    :color  'black ))

(setq edge-bonding-family
   (make-instance 'family :family-name 'edge-bonding-family
    :color  'blue ))

(setq tone-bonding-family
   (make-instance 'family :family-name 'tone-bonding-family
    :color  'red ))
```

Every scout will belong to one (in the current implementation) family such as the ones defined above. When proposing a new bond, the bond proposing function will register the proposing scout's family in the 'family' slot of that bond. As can be seen from the names of the example families above, families group together different scouts into functional properties. But before I delve into implementation details, let me illustrate the need for such a function.

Take, for instance, a simple tone in the input signal (see Figure 7-12, left). It *might* have a salient (i.e. large and clear) positive temporal derivative cue at its onset and a negative temporal derivative following at its offset. Notice the '*might*' in the previous sentence. That is the whole point of using context. In real-world sounds, such a tone *might* have these cues; it *might not* have them; or they may not be salient enough to be considered. They may even compete with other cues of the same sort that lie in the way (e.g. indicating amplitude modulation, or interference from other events from adjacent frequency zones etc.). Now, how can we tell if the two existing cues (a positive temporal derivative followed by a negative temporal derivative) signal the beginning and end-points of an auditory event? Well, in the ideal case, when there is no competing evidence, that is, when this is the only possible interpretation, then the problem is, of course, trivial. Generally, however we need to make use the redundancy of information across different elements. We should look for *different* types of evidence supporting the same interpretation. This brings us to the concept of mutual support for a proposed interpretation. Now, in the context of *The Ear's Mind*, interpretation of structures is left to emergent activities. Here I refer to the need of mutual support as a micro-tool for the settling of local fights between competing bonds. If each bond draws evidence for its existence only from the cues it relates, then the only way of settling a fight between two competing bonds is by considering properties local to the cues involved. However, an alternative way for settling such a fight could be to seek support from other (types of) bonds and cues. This type of support is mutual since it works both ways – the currently built bond may support a proposed bond. If the latter is built, it will support the former when it gets into problems (e.g. when a newly proposed bond competes with it).

Going back to the above example, the proposed bond relates two cues (a positive temporal derivative followed in time by a negative temporal derivative). If it gets support from other cues and bonds in the area, then this support should be used in the fight against competing bonds. In other words, mutual support should be (one of) the mechanism(s) which help to settle fights between competing structures. In Figure 7-12, pt1 has bonded with nf1 and nf1 with nt1 by scouts that seek to relate cues together which can form 'corners', or vertexes of auditory entities. They relate temporal and spectral cues that are 'well' located relative to each other. By appropriate functionality, the bonding pt1-nf1 and nf1-nt1 will provide support for building the bond pt1-nt1 rather than the competing bond pt2-nt2 (the two bonds are competing here only for the interpretation of onset and offset points). Although the given example is simplified for illustration purposes, it is by inspecting preprocessed real-world input signals that the need for such capability was found. In many such cases, using the described support system could settle fights that would otherwise be blindly settled by internal features only. In this example, such a blind settling strategy would favour bond *pt2 to* be built based on distance measurements.

Figure 7-12: In the left figure: A positive temporal derivative cue pt1 (left) attached to both a negative temporal derivative cue nt1 (right) and a negative spectral derivative cue nf1 (top). On the right figure: The additional positive temporal derivative cue pt2 at the bottom has failed to attach itself to the nt1. Even though pt2 is closer to nt1 than the already attached pt1,pt2 has lost a fight against the conflicting bond between pt1 and nt1 for reasons of mutual support from bonds pt1-nf1, and nf1-nt1. The dashed spline indicates a lost fight. See discussion above.

So scouts now belong to families, and every (proposed or built) bond knows to which family it belongs. We now define the 'bunch' class. A bunch is just a set of cues interconnected by built bonds. Such bonds should, however, belong to families which take part in the mutual support system. In general, certain families work only locally (e.g. the proximity family). In *The Ear's Mind*, a bunch of cues is not a *group*. A group is a compound object on its own. Such an object receives its own descriptions just like any other object in the domain, which enables it to be a part of bigger structures. The bunch can be seen as a set of cues, which may be on their way (if enough evidence exists), to becoming an object. For bunches to function correctly, it was found necessary to set rules defining which bonds may be allowed to coexist with others and which should be seen as conflicting with other bonds. This is vital both for avoiding complex loop-like situations in bond behaviour. The rules states that

1. Only one built outgoing bond of a given type could be allowed to exist at any time.
2. Only one built incoming bond of a given type could be allowed to exist at any time.

For an illustration of these rules, see Figure 7-13. The lower right sub-figure shows that the same interpretation of a given situation can be achieved without the need of contradicting the above rules. The existence of these rules, however, enables *The Ear's Mind*'s agents to work more locally and consistently, avoiding run time bug-like behaviour (i.e. unpredictable behaviour that arises in complex situations).

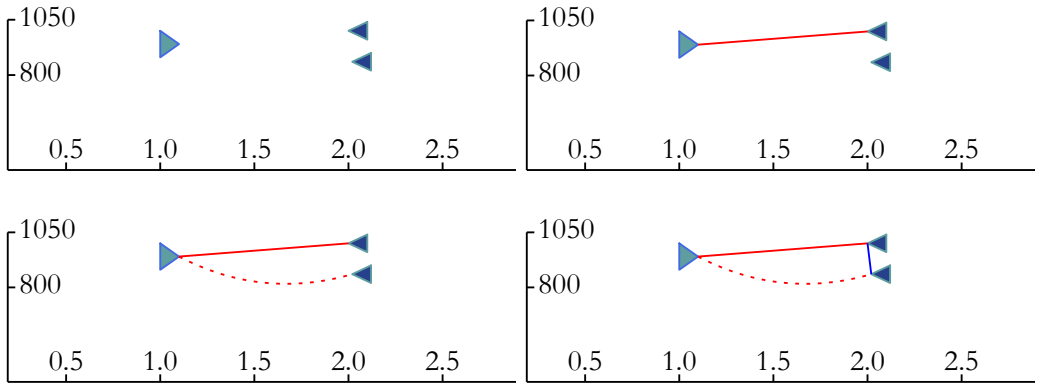

Figure 7-13: Example of one of the bonding rules for avoiding looping complexity

The bunch is defined as follows:

```
(defun make-bunch (family other-bunches bond-list cue-list
                   &aux new-bunch)
;;This returns a new bunch instance. Each of the input argument
;;initialises the corresponding slot.
;;this bunched may contain more than one family of bonds.
;;a family is a grouping of different types of bonds which shares
;;functional or organizational properties.
;;there is no limit to the number of families
;;
  (setq new-bunch
   (make-instance 'bunch
              :family family
              :other-bunches other-bunches
              :bond-list bond-list
              :cue-list cue-list
              )))
```

Each cue has 'outgoing-bonds' and 'incoming-bonds' slots, which are initialized to a hash-table with of bonds organised by families. In addition, a 'proposed-bonds' slot exists which lists all bonds currently proposed (but not built). Methods handling these slots were defined (for doing things such as adding and removing proposed or built bonds from slots, searching for bonds of a given family, maintaining the hash tables etc.). Additionally, functions were defined for providing the basic bunch functionality. For example, here is a list of some of these functions, with short descriptions:

- return bonds with same family and category which go out from the same cue but into a different cue than the one given
- return bonds with same family and category which go into the same cue, but from a different cue than the one given
- return a list of all the cues at the other ends of all the bonds connected to the given cue
- returns Boolean True if two cues are interconnected by any continuous chain of built bonds

- returns all the connected set of cues interconnected by any continuous chain of built bonds to the given cue
- add (or remove) given cue to (from) given family in the given bunch. If family does not exist, create a new one.
- merge the bunch of from-cue into the bunch of to-cue and delete the former bunch
- break a bunch into two separate bunches (following a removal of a bond that served as the only interconnection between otherwise two separate bunches).

The following 'fight-bond' function checks out the support that a newly proposed bond gets from existing bonds versus the support that the already built bond gets. It returns the result of this fight. 'won' will result in removing the built bond, and building the proposed one. 'lost' will result in cancelling the proposed bond, which means that the built bond lives on. Support score counts only the number of different families supporting the same bond rather than the number of cues involved in each family. If the bonds tie as far as support goes, it remains up to the 'solve-tie' function to decide what to do (e.g. using other measurements, toss a coin, favour the older structure etc.).

```
(defun fight-bond (proposed-bond competing-bond &aux diff)
;;compares the support given to proposed-bond and to competing-bond.
;;returns either 'won or 'lost. won and lost indicate the status of
;;the proposed bond following the fight.
;;
;;print to terminal fight details:
(format t "in fight-bonds, existing bond attached from :~a to ~a~%"
  (:pname (:from-obj competing-bond))
  (:pname (:to-obj competing-bond)))
;;
;;since the competing bond is already built, it gets support from its
;;own family already. this 'unfair' advantage will be countered
;;by adding 1 to the support score of the proposed bond.
;;function supporting-diff computes the difference in support between
;;the two bonds.
;;
  (setq diff (+ 1 (supporting-diff proposed-bond competing-bond)))
    (cond
     ;;if no difference in support, then solve the tie
     ((= diff 0) (solve-tie proposed-bond competing-bond))
     ;;if difference > 0, return 'won
     ((> diff 0) 'won)
     ;;if difference < 0, return 'lost
     ((< diff 0) 'lost)
     ;;otherwise return 'error
     (T (error "in fight-bond, reached end of cond."))))
(defun supporting-diff (bond1 bond2)
;;returns the difference between the nr of supporting families of the
;;first and the second bonds
;;
 (- (length (supporting-families bond1))
    (length (supporting-families bond2))))
```

7.5.9 Implementing the Preprocessor

At I have already mentioned, experimenting with models such as *The Ear's Mind* will best be done using two equally important strategies. Each of these offers unique opportunities for investigating the proposed model's behaviour and constraints. A model that offers both experimentation strategies will allow us to learn more about the domain than we would otherwise be able to. The first experimentation method makes use of symbolic definitions of artificial arrangements of primitive elements in the input domain. Translated into *The Ear's Mind*'s domain, it says, that this method *requires* input from *non* real-world sounds. Moreover, it is not equivalent to passing an artificially synthesised sound through the preprocessor. The idea is to arrange a selection of primitive elements for investigating the model's behaviour in idealised circumstances. One is able to test, for instance, the model reaction to temporal derivatives only, or a simple combination of specific types of cues in specific relative locations. Through such experiments, we do not only learn of the nature of our model (which, having an emergent nature is difficult to analyse), but get the opportunity to tune distinct properties such as the behaviour of individual agents, or zone definitions. The second experimentation method requires a setup for real world input to be fed into the model. Such capability should never be postponed to future expansion phases since the model's behaviour as a whole (or the behaviour of individual agents etc.) has to be designed gradually with observations of the model's real-world behaviour. Now, real-world input does not necessarily mean using full-fledged compound sounds from cocktail parties or construction sights. It means the whole range from 'clean' recordings of simple auditory events with all sorts of properties (percussive sounds, voices, granular sounds etc.) through simple sound combinations, to the above 'really' real-world sounds. Even if the model is 'not yet' ready for dealing with real sounds, it is vital to use them in early stages since such experiments keep us from designing a one trick model assuming that trick will provide solutions to all our problems. It is by working with real sounds, for instance, that I have concluded that no single (or even a set of just a few) preprocessed cues will suffice to approach even simple auditory capabilities. When testing preprocessing schemes in real world situations, one is consistently confronted with missing cues, sound artefacts (which produce 'wrong' cues), misleading cues, etc. By studying such situations, one can investigate what specific types of cues work and when they fail, and appreciate the role of cue redundancy.

The Ear's Mind was designed from the start to minimise the role of preprocessing so that no interpretation steps would be unintentionally skipped and set in stone by the preprocessor. Consequently, *The Ear's Mind* allows any preprocessor to analyse the input signal and produce a list of cues containing cue type, location and any other properties needed for a cue's definition. *The Ear's Mind*, at the other end of the symbolic interface will have to 'understand' the preprocessing list, and to have Codelets and structures that know how to work with such cues. This arrangement simplifies the extension of the model with new types of cues or preprocessing algorithms. Since communication between preprocessor and *The Ear's Mind* takes place through lists of symbolic cues, it is straightforward to use any of the two experimentation methods described above. For real-world sounds, we can feed input through the preprocessor, and produce (human-readable text) lists defining each of the cues we want to have. These lists are read by *The Ear's Mind* and the run begins. For any artificial arrangement of cues, we can manually construct lists in the same format used by the preprocessor engine. *The Ear's Mind* does not know in which mode of experimentation it runs. It just runs on different input files.

In what follows, I describe the preprocessing algorithm I have used for producing the current set of implemented cues. This set is by no means sufficient for a fully functioning auditory scene analysis engine. Although only a few types of cues are defined here, many more can be defined in principally the same way. In the current phase of *The Ear's Mind*, the preprocessor was written using Matlab, which enables quick scripting of signal-processing algorithms. The cues produced are:

- positive temporal first derivative
- negative temporal first derivative
- positive spectral first derivative
- negative spectral first derivative

These cues are produced from the power spectrogram as described below. Throughout the discussion, I use a fragment of sound example nr. 3 from the demonstration CD of auditory scene analysis (available with Bregman 1990) to exemplify the different signal processing steps.

The text file containing cues for *The Ear's Mind* is produced by calling Matlab script 'wav2em'. The script first enables us to open a sound file (currently in .wav format) to be bound to variable 'sig'. Although principally, any (sufficiently high) sampling rate could be used, I have used rates of 8kHz and 16kHz for input files, which seemed to work fine for the implemented cue types. Variable 'fs' contains the sampling frequency of the read file and is used later on. Functions uigetfile (GUI for browsing and choosing files) and wavread (for reading a .wav sound files) are standard Matlab functions.

```
%choose a file with gui
cd('c:/laptop/matlabcode/asa');
[FileName,PathName] = uigetfile('*.wav');
%open wav file, sound should be in mono format
file=strcat(PathName,FileName);
[sig,fs]=wavread(file);
```

Now, the complex spectrogram is computed by calling 'complexsp' with appropriate parameters. Currently, these are 2ms shift between successive windows, 32ms length of window (using the Hanning function), and zero padding to 512 fft length.

```
%compute a complex spectrogram
%sp's columns contain the result of one windowed fft each.
%if not supplied, the parameters default to:
% fs=8khz, 64ms hanning window, 8ms window shift
%and zero padding up to 1024 fft length

shiftms = 2;
sp = complexsp(sig,fs,32,shiftms,512);
```

Function complexsp computes the power spectrogram in a standard fashion:

1. compute the number of samples per window
2. compute the number of window shift size in samples
3. match the size of the fft window to a power of two (for effieciency)
4. make hanning window the size of winsamps
5. for all frames do
    a. scalar multiply the signal's window range by the window function
    b. compute the fft of the windowed signal
6. return spectrogram

An image of the absolute value of the returned spectrogram can be seen in Figure 7-14. The example is taken from asa CD track 1.

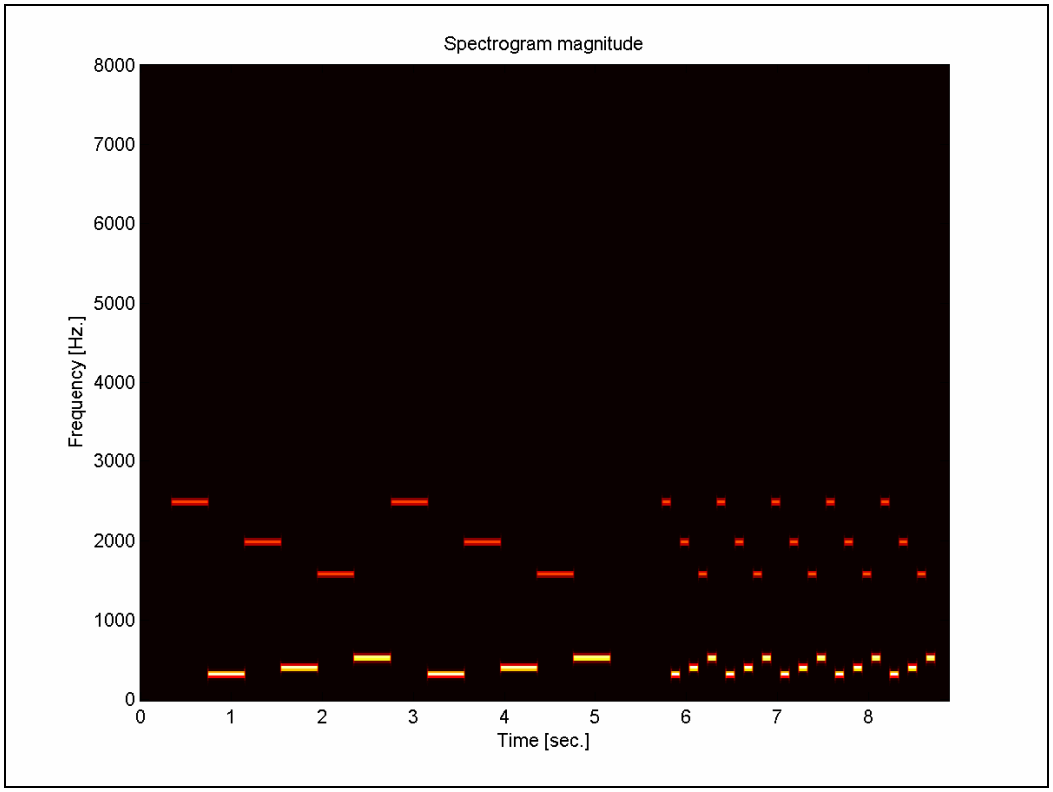

Figure 7-14: Preprocessor implementation stage 1

The complex spectrogram matrix, sp, is returned as a freq by time matrix. It is then normalised to a reference point relative to 0 dB, taking -200 as a minimum sound level.

```
[freq,time]=size(sp)
index=1:time;
sp(:,index)=max(20 * log10(sp(:, index)/1024), -200);
% Normalization to the reference sound pressure level of 0 dB
Delta = max(max(sp));
sp = sp -  Delta;
```

Next, the spectrogram is clipped at -30 dB. The result is shown in Figure 7-15.

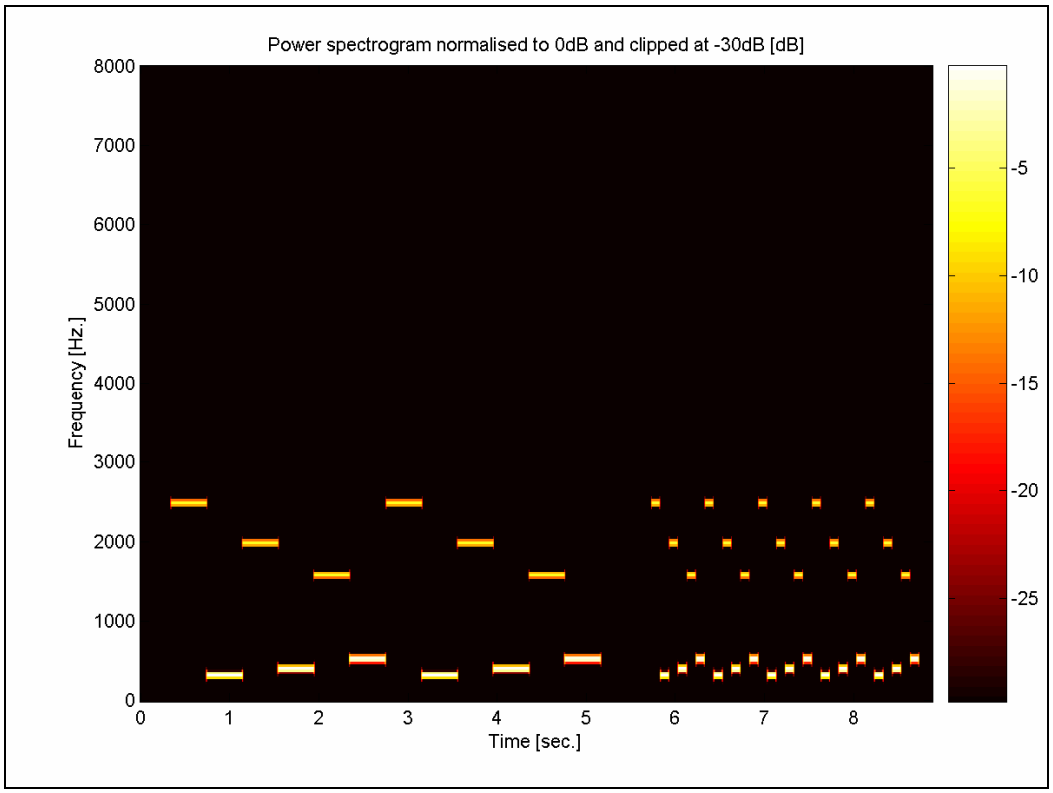

Figure 7-15: Preprocessor implementation stage 2

```
res=sp>-30;
sp30=res.*(sp+30);
```

Now, a difference approximation of the first derivative is computed separately in the time and frequency directions. Time derivatives are called td, and frequency derivatives are called fd.

```
%compute the time and freq derivative approximations
t=2:time-1; f=2:freq-1;

td=zeros(freq,time-1);
fd=zeros(freq-1,time);
td(:,t)=sp30(:,t+1)-sp30(:,t-1);
fd(f,:)=sp30(f-1,:)-sp30(f+1,:);
```

This results in time and frequency first derivative matrices, which are plotted in Figure 7-16 and Figure 7-17, respectively.

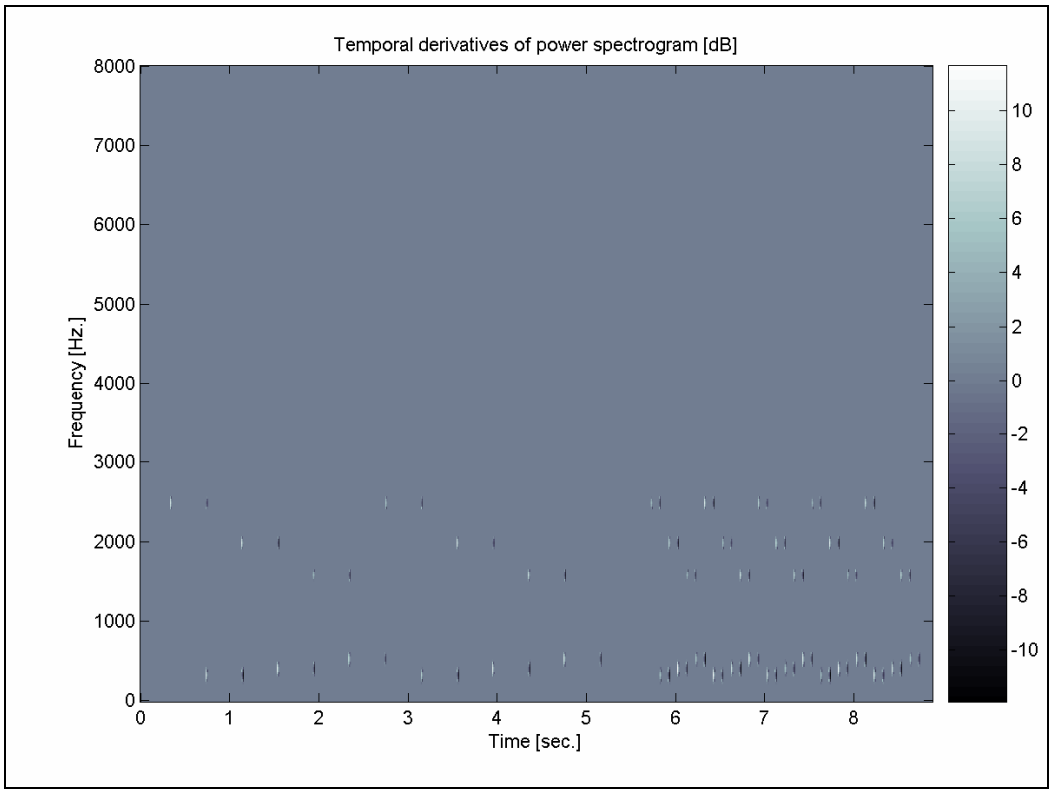

Figure 7-16: Preprocessor implementation stage 3

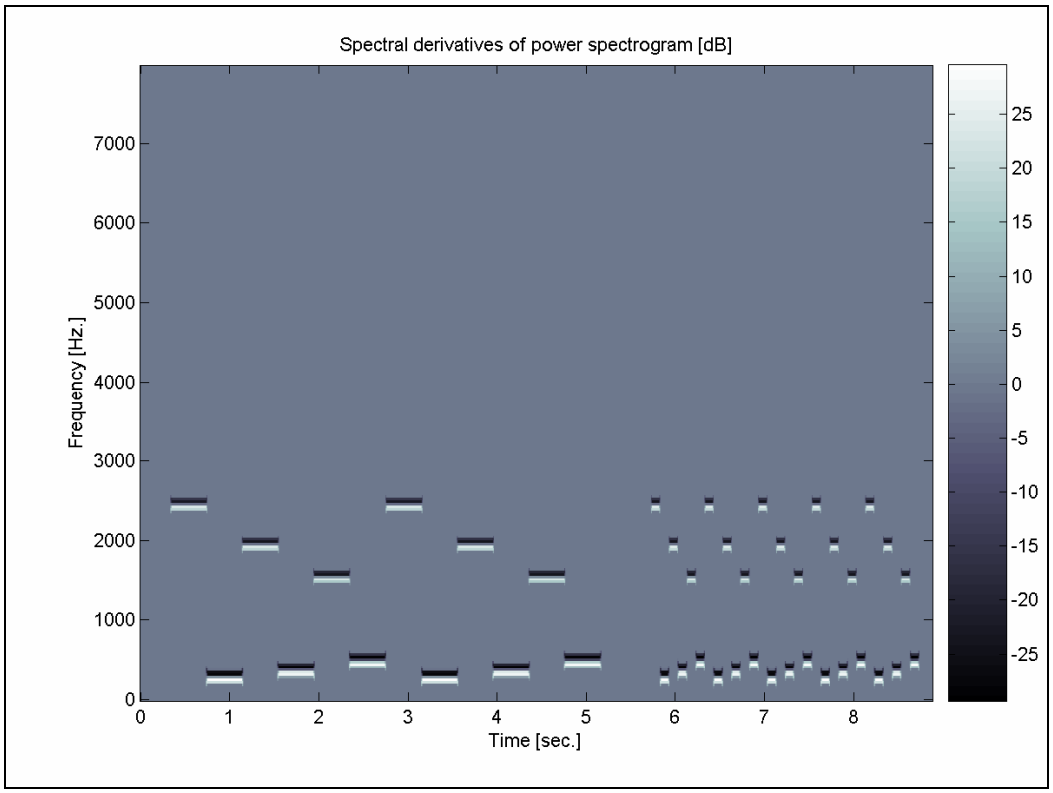

Figure 7-17: Preprocessor implementation stage 4

In the current implementation, all derivatives greater than 3dB (for positive derivative cues) or smaller than -3dB (for negative derivative cues) are included in the analysis file. Four binary matrices are prepared: postd and negtd for positive and negative time derivative matrices, and posfd and negfd for positive and negative frequency derivative matrices respectively. See figures 7-18, 7-19, 7-20 and 7-21.

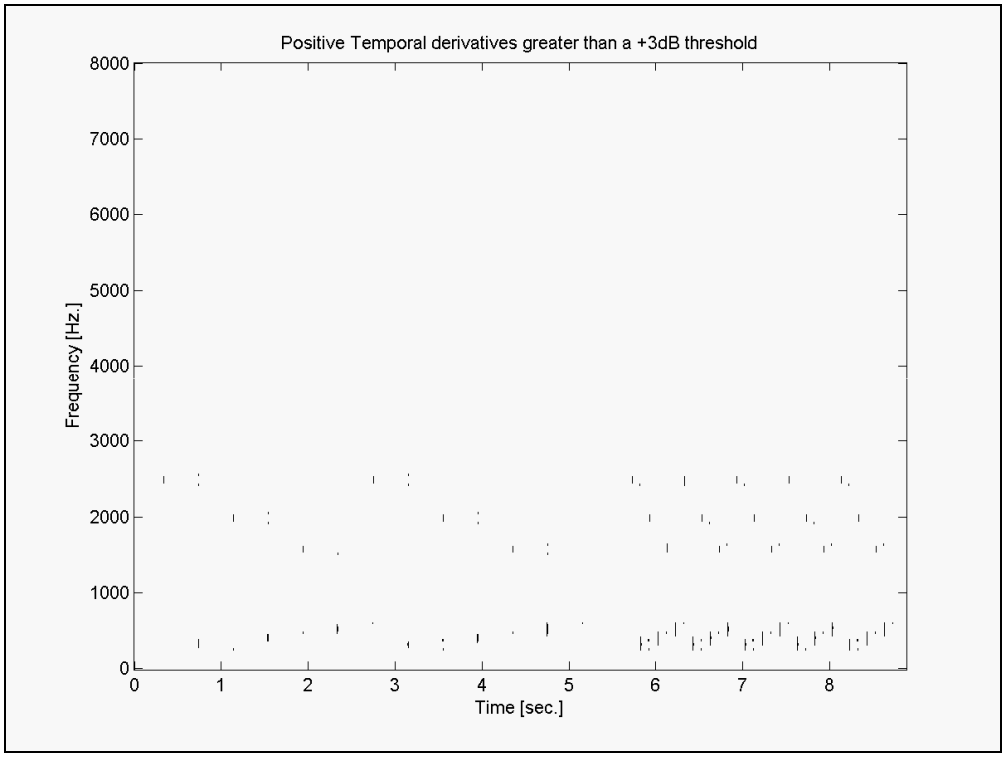

Figure 7-18: Preprocessor implementation stage 5

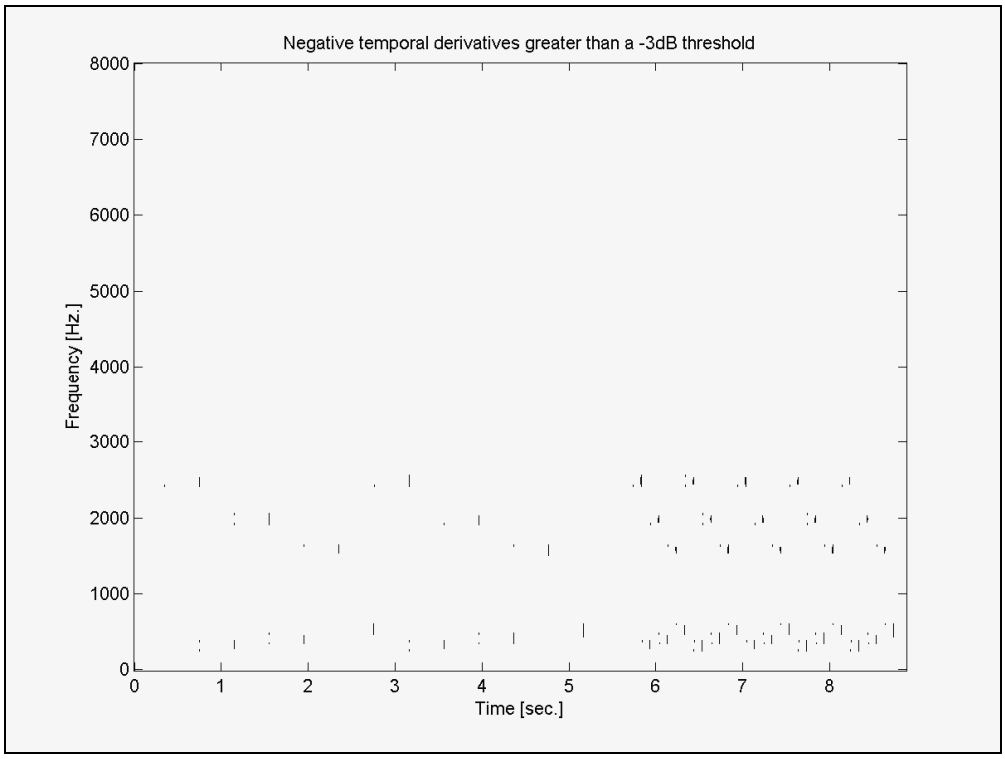

Figure 7-19: Preprocessor implementation stage 6

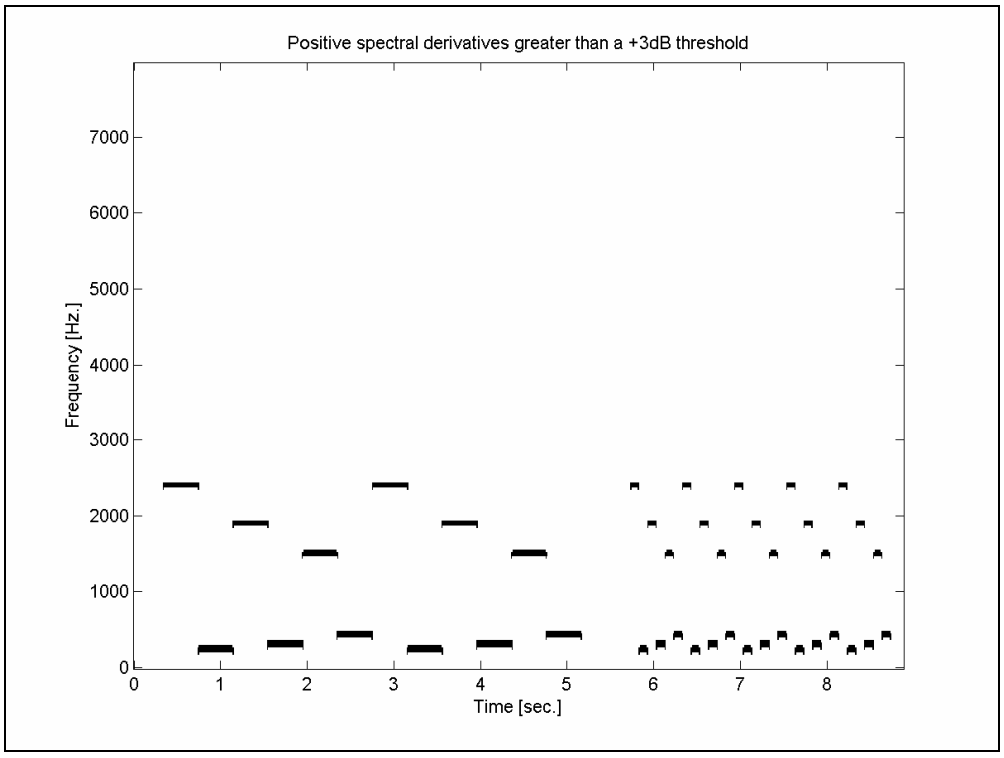

Figure 7-20: Preprocessor implementation stage 7

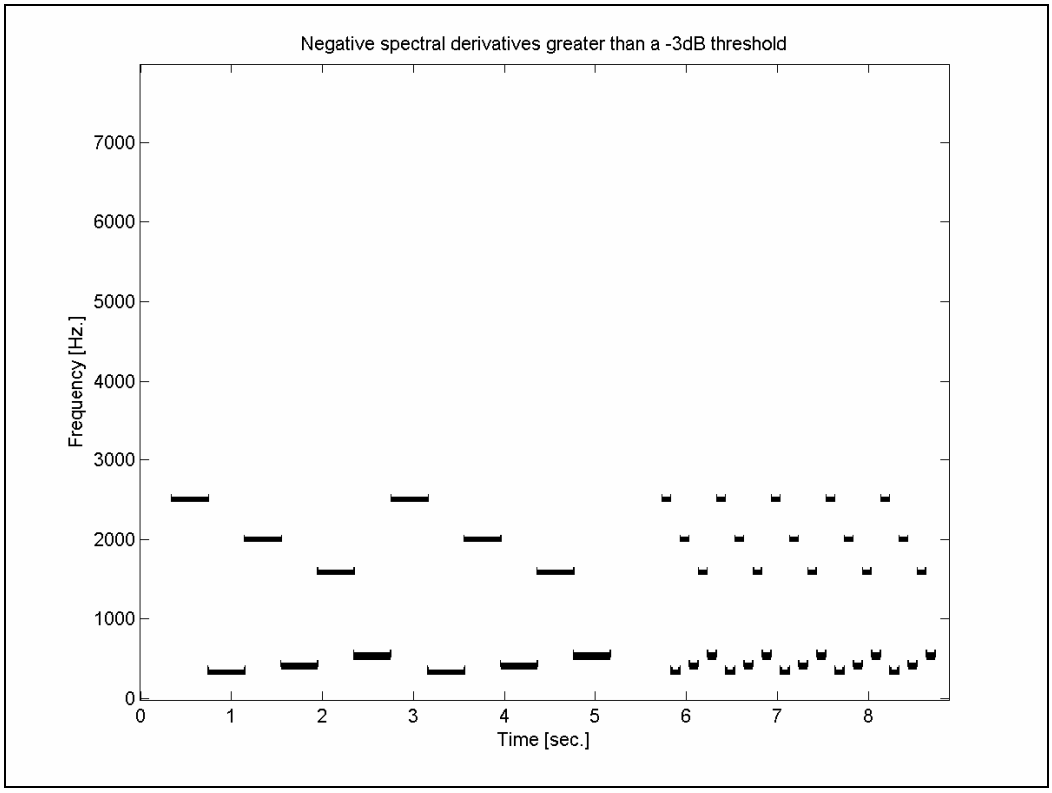

Figure 7-21: Preprocessor implementation stage 8

```
%compute binary images values are greater than 3dB
threshold=3;
postd=td>threshold;
negtd=td<-threshold;
posfd=fd>threshold;
negfd=fd<-threshold;
```

We now have four binary matrices with pixels of value one denoting retained salient derivative points. We now group together all connected the 1's in either vertical or horizontal directions. Each group of interconnected pixels is assigned a unique id number. This operation is carried out by the 'cue' function.

```
postd=cue(postd);
negtd=cue(negtd);
posfd=cue(posfd);
negfd=cue(negfd);
```

The cue function gets a binary two-dimensional matrix and outputs a matrix of the same dimensions in which each pixel contains the id number of the interconnected (vertical or horizontal only) group to which it belongs. The function executes the following steps:

1. zero pad the matrix
2. negate the matrix to reserve all positive numbers for id's. call a pixel with a value of -1 "on".
3. set new id number to 1
4. repeat for the frequency dimension from 2 to end
    a. repeat for the time dimension from 2 to end
        i. if current pixel is on then
            1. check pixel above
            2. if pixel above is non-zero then
                a. set id of current pixel to that non-zero value
                b. check pixel on the left.
                c. if the value of the left pixel is non-zero, and it is different than the value of the pixel above, then
                    i. set all the pixels in the matrix having the same value as the pixel on the left to the value of the pixel above.
            3. otherwise, if pixel on left is non-zero, then
                a. set id of current pixel to that non-zero value
            4. otherwise, set the pixel to the value of the new id number
            5. increase new id number

For a graphical representation of the above algorithm, see flowchart in Figure 7-22.

```
function map=cue(map);
%produce cues for The Ear's Mind.
%map should be a two dimensional binary matrix.

[freq, time]=size(map);
map(:,1)=0;
map(1,:)=0;
map=-map;

id=1;
for f=2:freq
   for t=2:time
      if map(f,t)==-1
         above=map(f-1,t);
         left=map(f,t-1);
         if  above
            map(f,t)=above;
            if and(left, left~=above)
               map(map==left)=above;
            end
         elseif left
            map(f,t)=left;
         else
            map(f,t)=id;
            id=id+1;
         end
      end
   end
end
```

Now, create a file for writing the Lisp code. This text file will contain the appropriate Lisp code for defining the cues in *The Ear's Mind*. The file is created with the same name as the opened .wav file, with .lisp extension. Next, for each type of cue, function 'edges' is called with the following variables:

- the resulting matrix from the cue function (containing groups with unique groups id's)
- the original sampling rate and window size of the computed spectrogram (used for calculating absolute time and frequency coordinates for each cue)
- the file handle (for appending code in the file)
- the name of the Lisp function which should be called for defining the given type of cue.

The edges function will create one line of Lisp code for each cue. Each group (identified by its unique id number) bigger than a predefined minimal dimension, will form a single cue. Smaller groups are discarded. Finally, the file is closed.

```
fid = fopen(strcat(file(1:end-3),'lisp'),'w'); %open (new) file with mode write
%this mode erases all existing contents if file already exists. catch file opening errors:
if fid==-1 disp('error opening file for writing in function edges');
end

edges(postd,fs,shiftms,fid,'make-pos-time-cue');
edges(negtd,fs,shiftms,fid,'make-neg-time-cue');
edges(posfd,fs,shiftms,fid,'make-pos-freq-cue');
edges(negfd,fs,shiftms,fid,'make-neg-freq-cue');

fclose(fid)
```

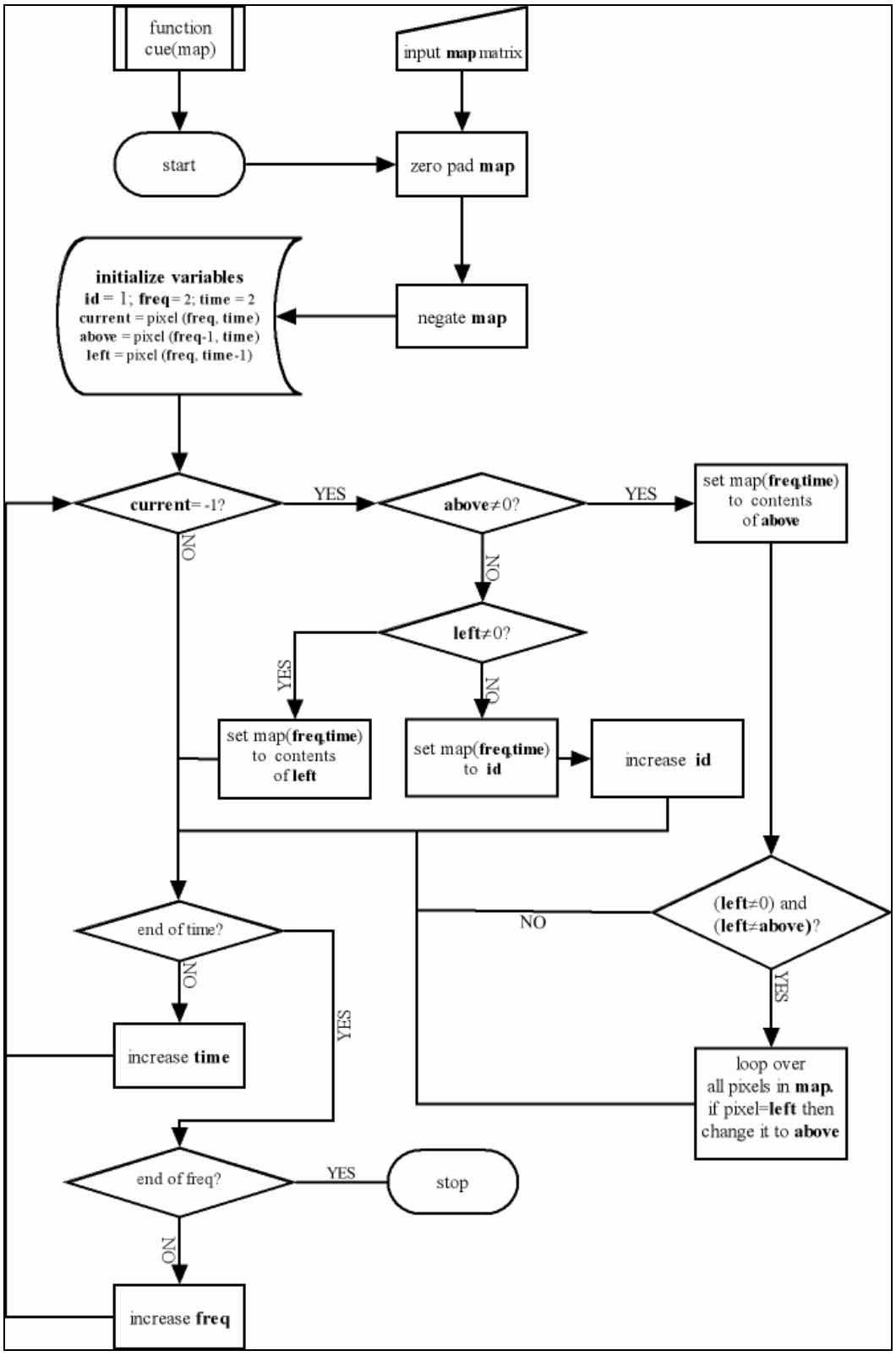

Figure 7-22: Preprocessor implementation flowchart of the cue function

Function 'edges' is implemented in the following way:

1. calculate some initial constants
2. create two indexing matrices, one for frequency and one for time measurements. Each pixel in the indexing time (or frequency) matrix gets the value of its time (or frequency) index
3. start with group id=1, and loop over all id number. For each id, do:
    a. create a binary matrix containing 1's only for pixels of value equals current id.
    b. Set tmax, tmin, fmax, and fmin to the maximum and minimum time and frequency indexes of the non zero elements of the binary matrix created in step 3.a. by using the index matrices from step 2.
    c. Check the indexes from 3.b. to see if the group is large enough using the constants for minimum sizes in the time and frequency directions
    d. If group is large enough, create it:
        i. Depending on the Lisp command parameter, append text to file containing the:
            - Lisp function call for creating the cue
            - A name for the cue, indicating the type and id number
            - The relevant coordinates for cue creation (e.g. absolute start and end of frequency and time locations)
            - Directive to attach the appropriate description type to the cue (e.g. positive temporal derivative)

    e. Otherwise (group is not large enough), do nothing
4. continue with next id from step 3.
5. return from function.

The list below contains some lines from each cue type in the file created using the wav2em script.

```
;;positive temporal derivatives:
(make-time-cue 't-cue-1 2447.05882353 2572.54901961 0.33400000 :width
0.01000000 :desc 'pos)
(make-time-cue 't-cue-2 2447.05882353 2572.54901961 2.74600000 :width
0.01200000 :desc 'pos)
(make-time-cue 't-cue-3 2447.05882353 2572.54901961 5.72400000 :width
0.01000000 :desc 'pos)
.
.
.

;;negative temporal derivatives:
(make-time-cue 't-cue-1 2447.05882353 2572.54901961 0.75600000 :width
0.01200000 :desc 'neg)
(make-time-cue 't-cue-2 2447.05882353 2572.54901961 3.16800000 :width
0.01000000 :desc 'neg)
(make-time-cue 't-cue-3 2447.05882353 2572.54901961 5.83600000 :width
0.01200000 :desc 'neg)
.
.
.
```

Finally, Figure 7-23 shows the graphic server of *The Ear's Mind* after reading and drawing all cues from the file prepared by the wav2em preprocessor. Now *The Ear's Mind* is ready to release its agents on this input landscape. See for details on experimentations carried out so far in chapter 8.

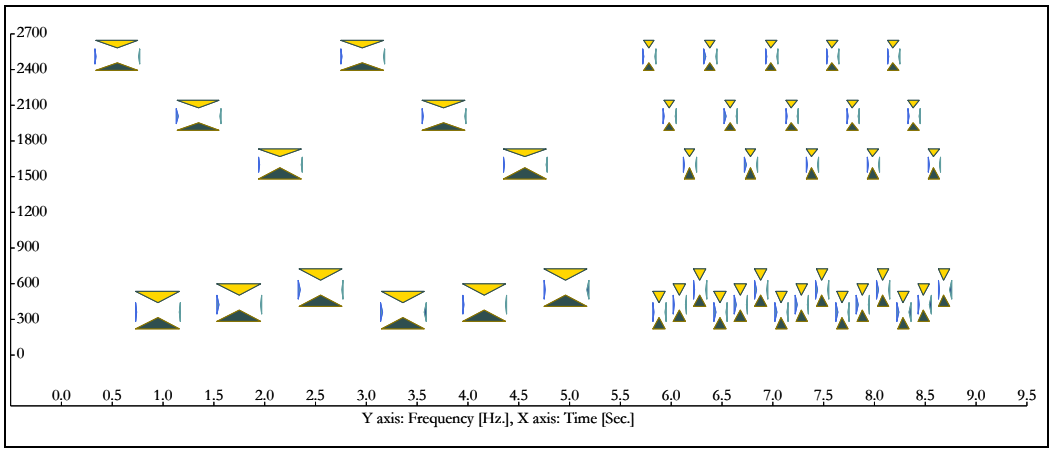

Figure 7-23: Graphics server display of cue definition file as received from the preprocessor

7.6 Implementation assessment

In the previous section, I have shortly described the main implementation steps taken for the realisation of The Ear's Mind. In what follows, I review those steps from a technical point of view. Evaluation of The Ear's Mind itself as a proposed theoretical model is done in chapter 8.

7.6.1 Choosing Common Lisp for the implementation of The Ear's Mind

Using Common Lisp for *The Ear's Mind* was initially motivated by the need to port Copycat's source code and get it to run on modern computer systems. The original rationalization for choosing common Lisp as a target language was ease of porting due to syntax similarities between the Lisp dialects. Additionally, writing a compiler for translating between different dialects in Lisp, though not a trivial task is quite a straightforward process. I believe that the choice was indeed the right one. The compiler structure was simple since the Lisp reader (parser) itself was used for reading the original source code. Furthermore, since the ported and original code were kept correlated (with regard to both programme structure and general syntax), it was easy to compare the two versions, and to consult the original source code documentation.

Once Copycat was up and running in common Lisp, it seemed evident that *The Ear's Mind* should be implemented by altering the ported Copycat and supplementing it with the new modules for *The Ear's Mind*. The Lisp support for sophisticated macro creation proved especially handy for the development of the proposed model. The use of the macro enabled me to test different implementation strategies (e.g. for scout behaviour) by writing only a few lines of code for each version.

Since neither execution speed nor code efficiency were an issue for the initial testing phase, it seems that the flexibility of Lisp contributed to the rapid implementation and testing of the model. However, even when execution speed is an issue, there exist modern Lisp compilers that can produce code that runs at comparable speeds to c. The ability to implement and test individual functions using the Lisp interpreter prior to compilation greatly enhances the development cycle. Each individual function was first tested on its own with test input by the Lisp interpreter. The existence of the interpreter also enabled the use of input Lisp files (produced by the preprocessor) for creating cues at run time.

Several functions were quickly implemented to allow testing of their model behaviour. Functions, which slowed down the entire programme, were altered for efficiency, though only in extreme cases. Such were for example the case with the implementation of the time-frequency vectors were indexing was added for speeding up execution when searching for existing coordinates.

7.6.2 The tcl/tk graphics server

Tcl/tk was chosen to quicken the implementation of the graphic display. Since tcl/tk has many built in functions including support for multi-thread socket communication, plotting of primitive elements (e.g. lines, splines etc.) and user interaction (mouse and keyboard binding), it was left only to implement functions for computing positioning and for interfacing between the graphics server and *The Ear's Mind*. Additionally, tcl/tk offers such functions as postscript saving functions that further ease the implementation of a complete graphics server. The use of a socket interface will also allow future architecture freedom in case *The Ear's Mind* will be implemented as dedicated hardware etc. the use of the available graphics functions meant that virtually no time was spent implementing graphic primitives. This fact, combined with the portability of tcl/tk proved very successful.

The implementation of the graphics server allows the user to follow the behaviour of *The Ear's Mind* while it runs, asses the results, and save any number of predefined steps as postscript images (including the automatic creation of titles etc.) the server proved very important to the analysis and demonstration of *The Ear's Mind*. This usefulness of the graphics display for demonstration purposes seems to draw on the perceptual parallels between visual and auditory perception as suggested by the Gestalt school. We seem to be able to asses grouping tendencies with our eyes even when the images originate from auditory data. Although this ability is not absolute, it does result in effective evaluation of the model by visual means.

7.6.3 Implementing the Preprocessor

I chose Matlab for the implementation of the preprocessor due to its vast library of predefined functions (reading sound files, the signal processing toolbox, etc.) the process of writing Matlab scripts and functions is straightforward, and each function can be tested on its own. The use of predefined functions greatly saves implementation and testing time. The built-in functions for saving images and plots, and creating text files, made it easy to display signal processing results and create Lisp cue definition files for *The Ear's Mind*. Additionally, using Matlab enabled me to test several different signal-processing approaches for analysing input sound files and to reuse my own library of past implemented signal-processing functions.

8 The Ear's Mind: Performance

In this chapter, I include some of the preliminary tests with which I have challenged The Ear's Mind. These tests have to be seen in the right context for correctly evaluating their results. First, The Ear's Mind does not in any way 'know', that is, it is not prepared for the nature of input in advance, nor does it know the 'right' answers, or even what would constitute a right answer. Second, the tests show here are preliminary in the sense that the mechanisms responsible for emerging structures are tested separately as much as possible. This is entirely due to the current developmental stage of The Ear's Mind. The plan is to carry on the research in the same direction, if, and only if, such preliminary tests show promising results. Therefore, only few scout types were implemented, and their interaction was restricted to the basic cooperative/competitive actions as described earlier on in this work (see section on families and bunches). In addition, a proximity scout was implemented and tested to see how such grouping pressures would function in an emergent environment. Finally, since this work forms the first rapport of the proposed model, some attention is given to the behaviour of the preprocessor as well. All input sound fragments used in this chapter were taken from the compact disk accompanying Bregman's Auditory Scene Analysis.

8.1 Phase I tests and experiments

The tests are divided into two sections: in section 8.1.2 I present tests concerning emergent behaviour stemming from the local cooperation within families and bunches. In section 8.1.3, the competitive emergent behaviour of the implemented proximity scout is examined. But first, in section 8.1.1, some insight is gained if we observe the output of the implemented preprocessor.

8.1.1 Preprocessing

One of the principles I have argued for throughout this work is the need for emergent sensitivity to context rather than blind processing of input files. This does not only lay constraints on the implementation of the proposed model. The preprocessor seems to principally escape the usual demands put on it in other computer models for auditory scene analysis. This is because *The Ear's Mind* makes no use of 'smart' features in its input. Rather, the intension is to keep things as primitive as possible, and let *The Ear's Mind* do its work. Although preprocessing is never really trivial, all the preprocessor has to do is to produce lists of primitive features from the input sound files and hand these over to *The Ear's Mind*. Failures due to parameter misfit for some sounds should be handled by *The Ear's Mind* instead of by the preprocessor. The only compensation needed is to provide *The Ear's Mind* with lists of redundant features rather than going for a 'perfect' input. In this section, I try to exemplify my belief that no single preprocessing scheme could be found to work for every input, and that the only solution is to implement a preprocessor, which outputs feature lists at different sets of parameters. Such behaviour can be observed biological auditory systems as well. Therefore, for the preprocessing scheme in this work I considered simple spectrogram based analysis sufficient. It goes without saying that *The Ear's Mind* will accept any other preprocessing schemes and additional lists of

features, and let its agents fight for finding a solution to the puzzle. For implementation details, and examples of preprocessing stages, see (section implementation preprocessor).

When the input is simple, there may be a correspondence between salient points in the input sound the way humans perceive it and the preprocessed input features. In the spectrogram of Figure 8-1 below, we see that the input sound (consisting of simple pure tones produced artificially) is so defined that strong first derivatives in both the time and frequency directions correspond to the actual edges of the tones. This is a trivial case, and we may never rely on such correspondences in real-world sounds. In fact, real-world sounds are so complex that, I do not believe the solution to auditory scene analysis to lie at finding better preprocessing parameters or algorithms.

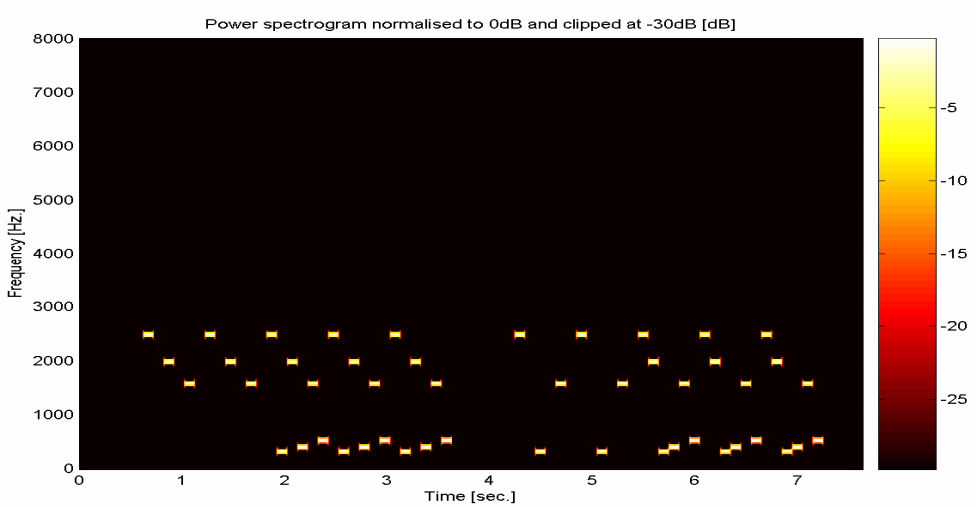

Figure 8-1: Spectrogram of artificially synthesized simple tones

Figure 8-2 the extracted positive and negative temporal derivatives are given. Due to the use of abrupt on and offsets, and to the absence of and other sounds or disturbances, the extracted features have a one to one correspondence with what we *hear*.

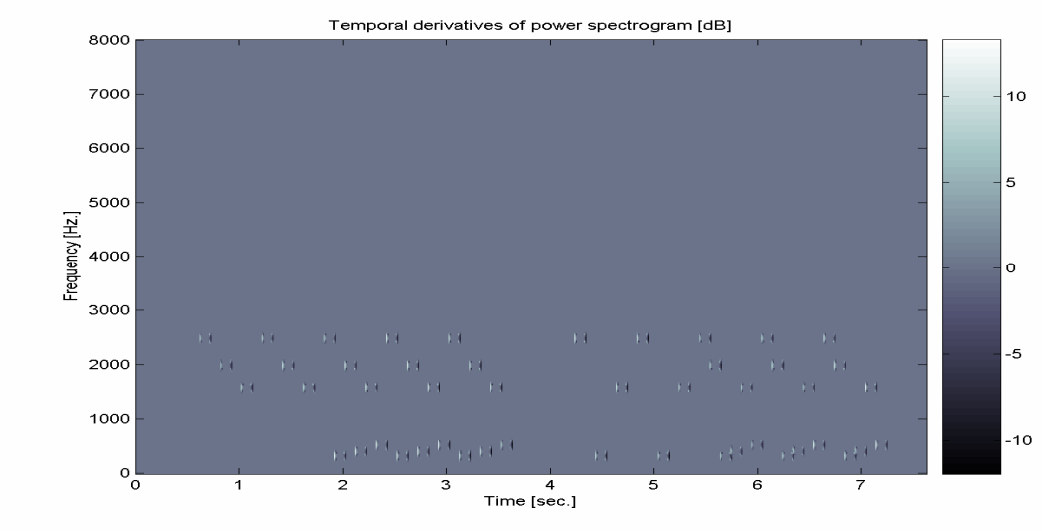

Figure 8-2: Temporal derivatives of artificially synthesized simple tones

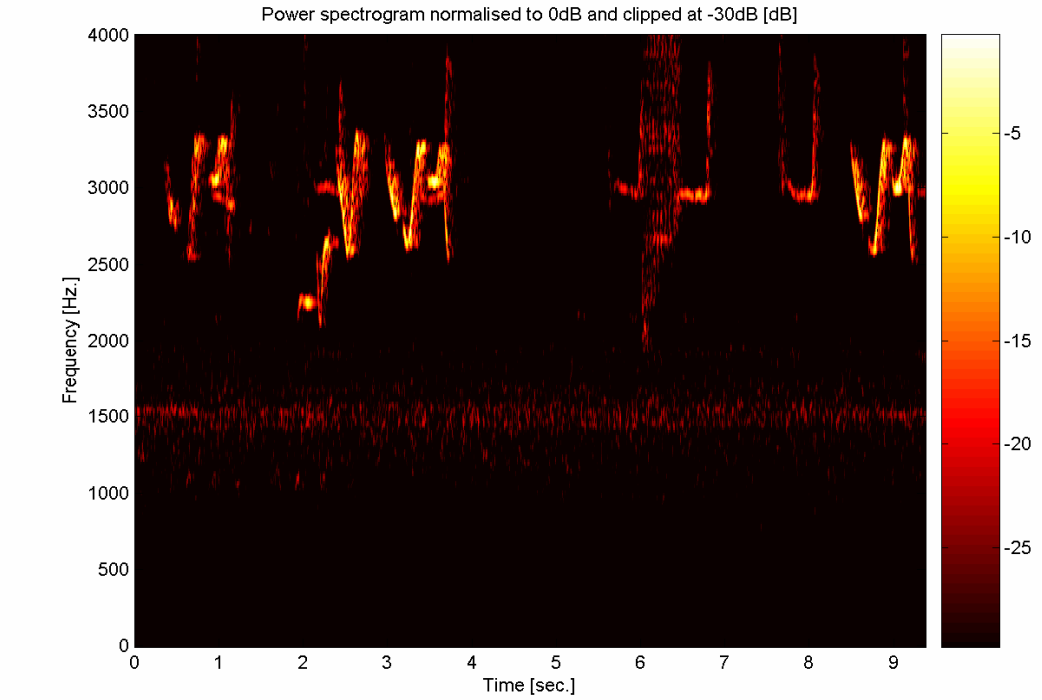

Figure 8-3: Preprocessing real-world sounds, a bird song in background forest noise

When the ear receives real-world sound, features cannot be put together correctly without context considerations. No preprocessing scheme could be used to solve this problem for all kinds of input. In such cases, no correspondence exists between physically related features in the

input and properties of perceived entities. Preprocessing is blind. For a simple real-world example, see the spectrogram of Figure 8-3 above. The spectrogram belongs to a recording of a song of a single bird in some background forest noise.

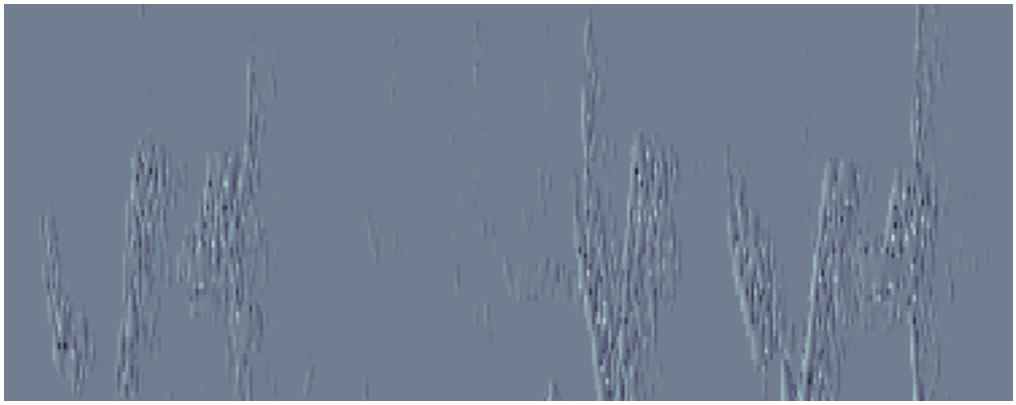

Figure 8-4: Preprocessing, detail view of temporal derivatives of the above bird song

Let us look at detail sections of a temporal (Figure 8-4) and spectral (Figure 8-5) derivative images computed from the above spectrogram.

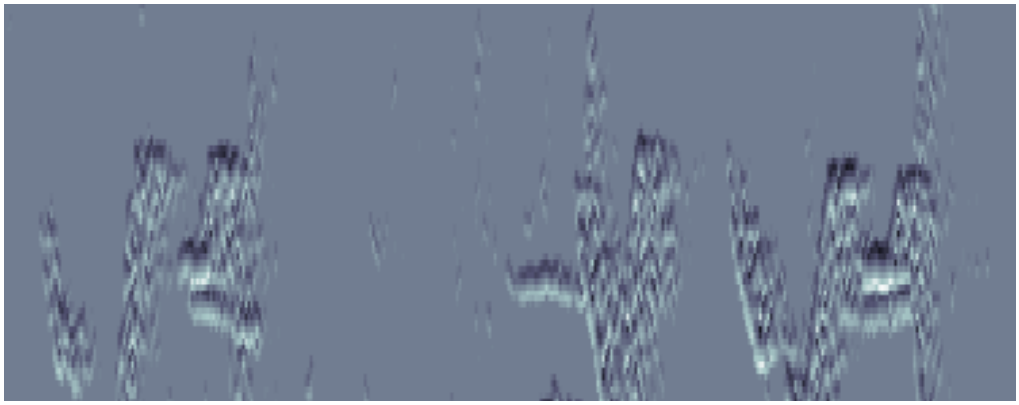

Figure 8-5: Preprocessing, detail of the spectral derivatives of the above bird song

It is apparent that many relatively strong derivatives exist in places that do not correspond to onsets and offsets the way we humans hear them. Actually, in agreement with the Gestalt school, our eyes have no problem in detecting the regions of sound that would be perceived as auditory entities by our ears. Needles to say, this is not always the case since sound has its own modal idiosyncrasies, but the point is that the quest for the 'right parameter' for detecting onsets is futile. Good news for designers of preprocessor algorithms, though.

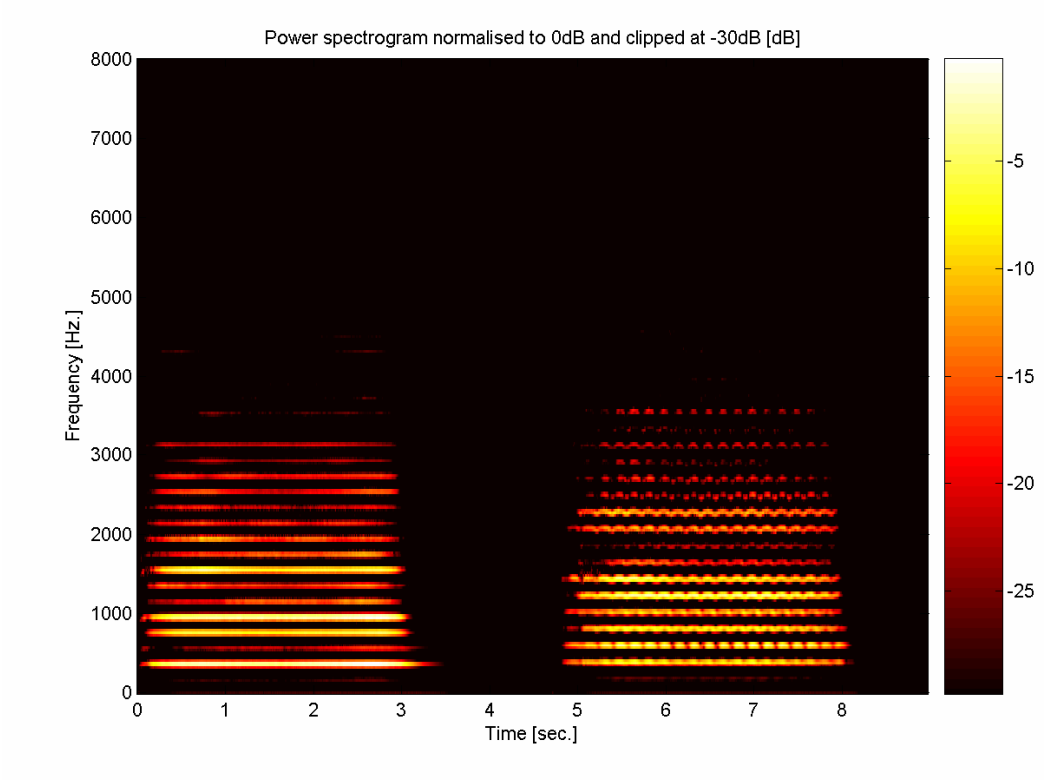

Figure 8-6: Preprocessing real-world sounds. Open G followed by a G sharp with vibrato on the violin

As a second example, I include the spectrogram of two violin notes played in a recording studio. The first note is an open G-string, while the second is a G sharp with vibrato (see Figure 8-6).

Now, when we compute the temporal derivative image from the spectrogram of the G sharp note, we get Figure 8-7.

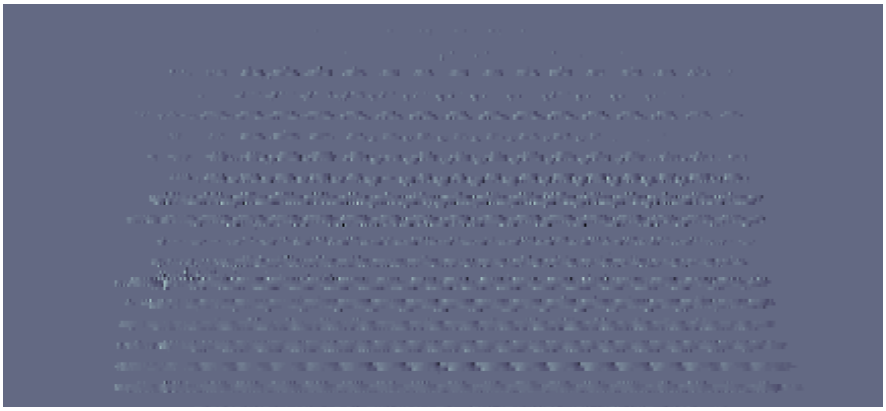

Figure 8-7: Preprocessing, detail temporal derivatives of a G sharp violin note with vibrato

Because of the vibrato played by the violinist, both frequency and amplitude modulations occur. Consequently, feature extraction in the usual blind sense breaks down, since there is no way of knowing (without consulting the context with other features and among the different harmonics of the note) if we should interpret this section as many separate short sound entities, or as a long note with frequency and amplitude modulation. In the simplest case, amplitude modulation will produce what may be interpreted as on- and offsets within the duration of the note, while frequency modulation will produce on- and offsets at different frequencies above and below the original frequency of each of the constituent harmonics. Notice also, that each harmonic shows a different sort of derivative behaviour, which would render any blind preprocessing scheme useless.

8.1.2 Context

For demonstrating the preliminary results of emergent context sensitivity using the concepts of families and bunches, I use track 1 of the auditory scene analysis demonstration compact disk. The preprocessing of this sound fragment was used to exemplify the different preprocessing stages in section 7.6.3. Figure 8-8 below depicts the cues received by *The Ear's Mind* when reading the cue list file.

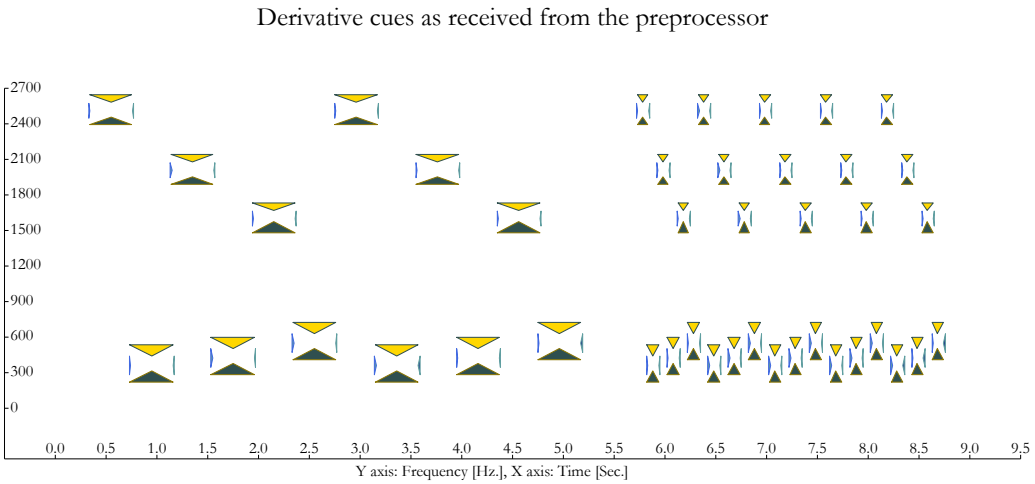

Figure 8-8: context test, track 1. Starting point

At this stage, the aim is to group the four types of cues positive and negative derivatives, in both temporal and spectral directions into 'good' groups. Such groups will form the basic entities, which would allow further grouping at higher levels of abstraction. For now, though, the task is to reach such 'good' groups by emergence of local activity, rather by any global processor or by using global grouping parameters. Such global parameters will fail here and certainly in real-world sounds since no absolute measures exist to what forms 'good' groups. In fact, *The Ear's Mind* does not even have any knowledge of the kind of groups we want. Instead, local agents land on two cues in the landscape of Figure 8-8 and attempt to build bonds between them. Take for instance the positive-to-negative-temporal-scout. It lands somewhere on a positive temporal cue (in the image portrayed by small triangles pointing to the right) and 'sniffs' for a negative temporal cue (triangles pointing to the left). Although its sniffing mechanism does prefer larger, closer cues, there was no attempt to define ideal distance or size parameters to ensure the

formation of good groups. Consequently, bonds are attached, which seem to contradict the desired result. In

Figure *8-9* below, the first bond is attached (encircled for clarity). It belongs to the edge bonding family, and bonds negative spectral cues to negative temporal cues using its sniffing zone to find those cues that fit such a vertex arrangement. In Figure 8-10 we see the landscape after some 35 frames of scout activity. Other bonds have been built, and the one encircled is attached between a positive spectral cue and a negative spectral one. Notice, however, that the bond seems to go against what we see as good grouping. It bonds two cues that are too far away in this case. Other bonds in the lower right region of this landscape also fail to 'know' the existence of 'better' grouping opportunities. Any attempt to correct this behaviour by tweaking the distance parameters will be futile for two reasons. First, different situations call for different parameters. There is no ideal distance parameter; even in this landscape, limiting the sniffing zone to shorter distances will fail to work on other regions. It is certainly not the case either, that bonding should always occur between the closest neighbours. Second, we have a sense of what constitutes here a good bond only because we use the context around the bond to evaluate it rather than judge it by any absolute measure. Consequently, the bond is built, and context pressures should emerge to steer grouping in the right direction.

Frame 2: a newly proposed bond is built.

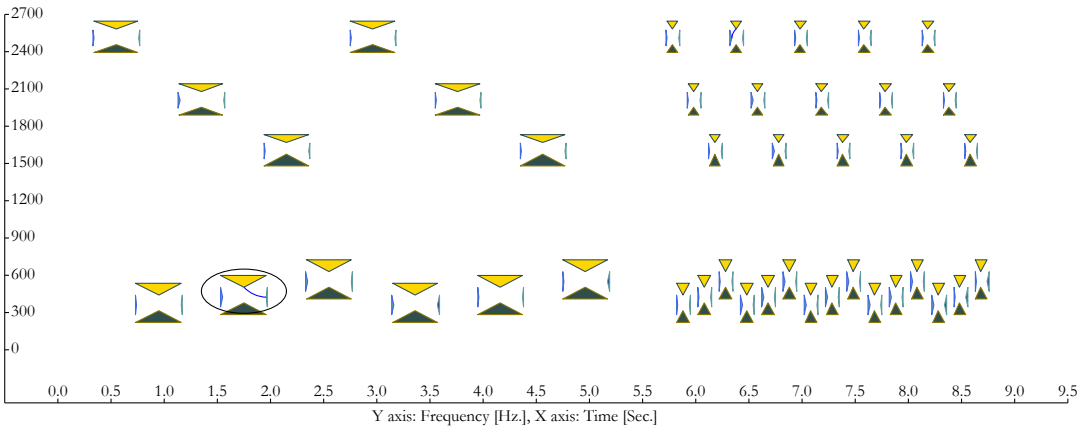

Figure 8-9: context test, track 1. Building bonds

Frame 35: a newly proposed bond is built.

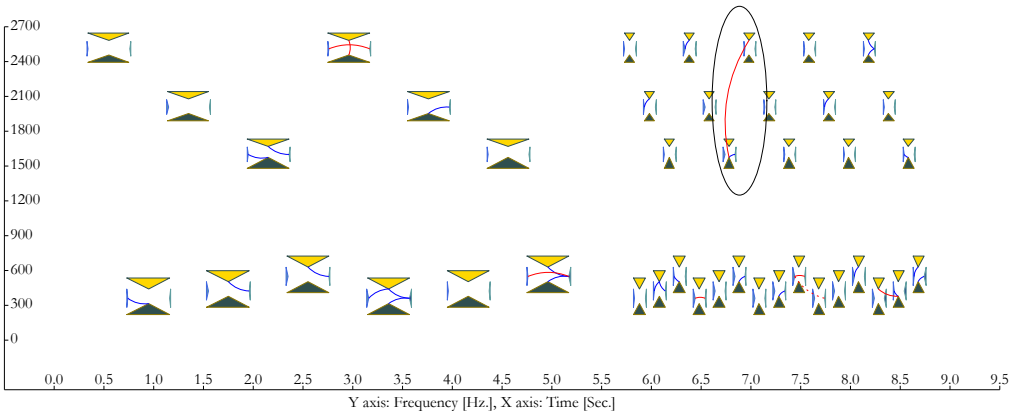

Figure 8-10: context test, track 1. Building bonds

In Figure 8-11, the newly proposed bond (encircled) is locally conflicting with an existing bond. Figure 8-12 depicts a detailed view of this situation. The proposed bond (encircled and drawn dashed), attempts to attach itself between a positive temporal derivative cue and a negative temporal derivative one. However, the negative temporal derivative cue is already attached by a bond of the same type to the cue directly on its left. This constitutes a conflict for the scout that proposed the new bond. It can only see locally those cues that resist its proposal, and will have to fight them. In the current implementation, the wining side will be allowed to build (or keep) its bonds; the bonds at the losing side will be destroyed. In this example, there is a reason for the existing bond to win by context, since the two cues it attaches are also attached by other bonds. The two cues in the figure are attached both by the positive to negative temporal bond, belonging to the internal-bonding-family, (whose bonds are coloured red), and indirectly by bonds belonging to the edge-bonding-family (coloured blue). Since the bonds of the two families share cues between them, they belong to the same bunch. Subsequently, the edge bonding family gives support for the internal-bonding-family (existing) bond. No support exists for the newly proposed bond. This is the advantage of redundant evidence. Different pieces of evidence are non-idealy grouped together by local agents. Their mutual support may however, signal a significant contextual reason to favour their view rather than the view of unsupported pieces of evidence. In this example, and for the reasons above, the proposed bond lost the fight. This is the reason why it is rendered with a dashed spline. Dead bonds are dashed to enable us to follow the cooperative/competitive activities as it occurs. Additionally, *The Ear's Mind* produces text reports on every fight detailing the reasons for fight results etc.

Frame 146: a newly proposed bond fights and loses against already built bonds

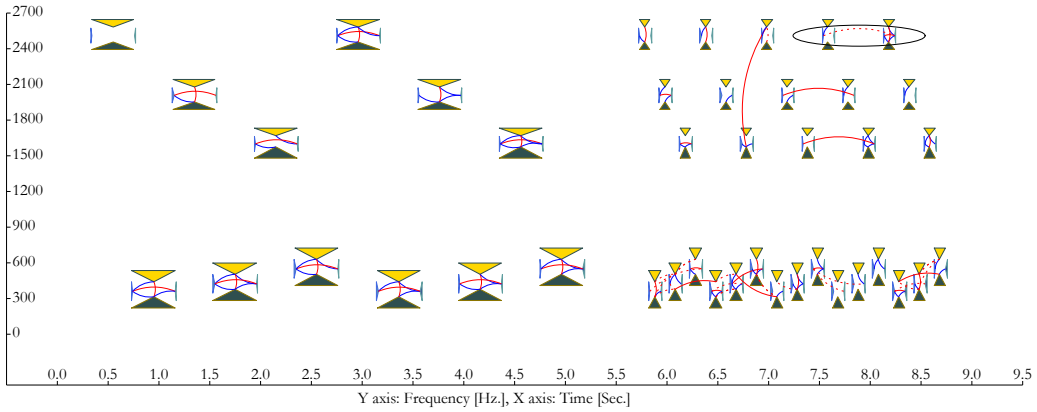

Figure 8-11: context test, track 1. Building bonds

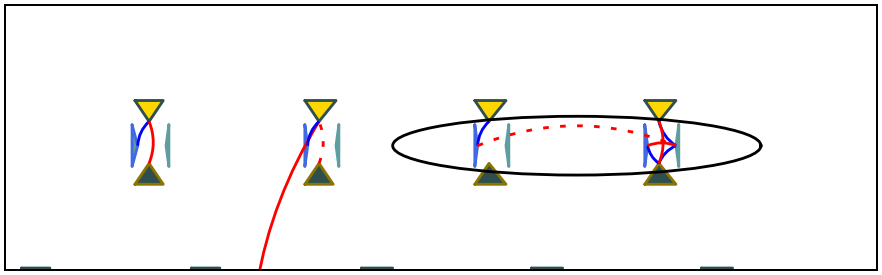

Figure 8-12: context test, track 1. Detail of mutual support in bond fights. See text for details

Another example of the working of families and bunches can be seen in Figure 8-13 below. Here, we can see that good groups have been built in the left region without resorting to fights. This result is by no means programmed, it just happens to fit the parameters of this specific region such as spacing and arrangement of cues. In the right region, we can see many dashed splines. The dashed splines are dead bonds that lost fights against other bonds. The encircled dashed bond is shown in detail in Figure 8-14.

Figure 8-13: context test, track 1. Building bonds

In the detailed figure, we can see that the proposed bond was attempting to attach a positive spectral cue (lower triangle pointing upwards) to a negative one (upper triangle pointing downwards). The two cues already belong to two different bunches, each with their supporting families. No conflicting bond exists since no cue of the proposed bond is attached to an already existing bond of the same type. The conflict here arises out of the context of the two bunches. Both bunches have already members of the edge bonding family. Both have top (triangles pointing downwards) edges and bottom edges (triangles pointing upwards). Notice, however, that only bonding by an edge bonding family makes a cue such as a positive spectral cue (upward triangle) an edge cue.

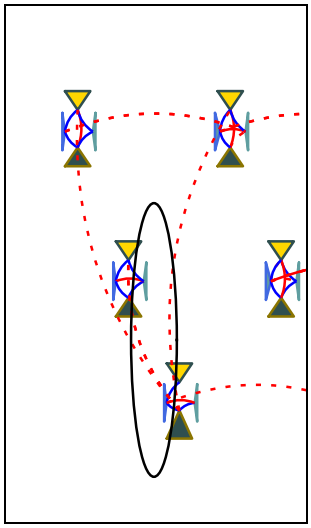

Figure 8-14: context test, track 1. Bunch resisting a conflicting bond. See text for details.

In this case, the families of the two bunches resist the proposed bond since the resulting group will have multiple conflicting edge cues. For these reasons, the proposed bond loses. It is then rendered in dashed spline. 1. The general guidelines behind the implementation of mutual support are:

- Lack of evidence does not give enough information either way. That is, the opinion of a single family will not have enough information to win fights.
- If two (or more) families support the same structure, then the structure has a good reason to be built.

Another case is depicted in Figure 8-15. By circumstances, the left group attached to the long bond is very well self-supporting. A newly proposed bond attempts to attach the left cue of the existing long bond to the negative temporal cue directly on its right. The proposed bond is thus conflicting with the existing bond (see Figure 8-16). In this case, the proposed bond gets full support from the families of the left group, and the existing bond loses. In this example, I have disabled the display of dead bonds for clarity.

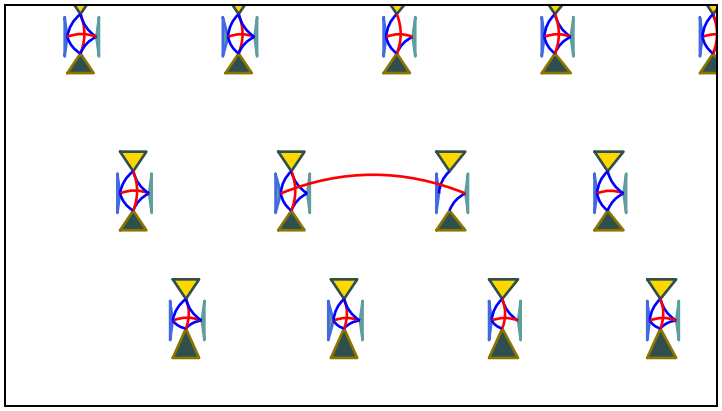

Figure 8-15: context test, track 1. Detailed view lower right region, see text for details

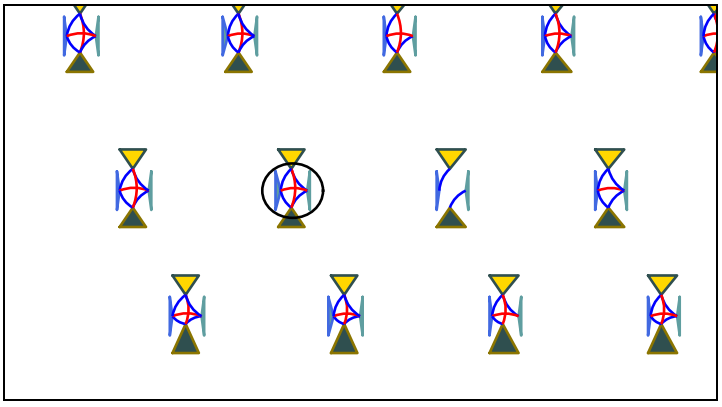

Figure 8-16: context test, track 1. Detailed view lower right region, see text for details

Finally, I include the following few figures to show the emergence of grouping by context from non-ideal local bonding activities. Figure 8-17 depicts the landscape in the midst of activity, showing only the dead bonds so far. Figure 8-18 shows the same moment in time but displays only the existing bonds. Many dead bonds seem to be wrong. Likewise, most existing bonds seem to make contextual sense.

Frame 250: a newly proposed bond fights and loses against already built bonds

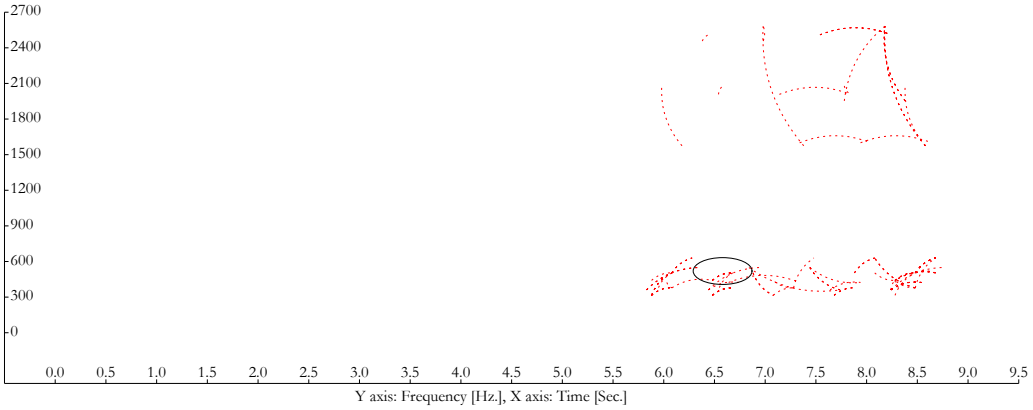

Figure 8-17: context test, track 1. Display of dead bonds only

Frame 250: a newly proposed bond fights and loses against already built bonds

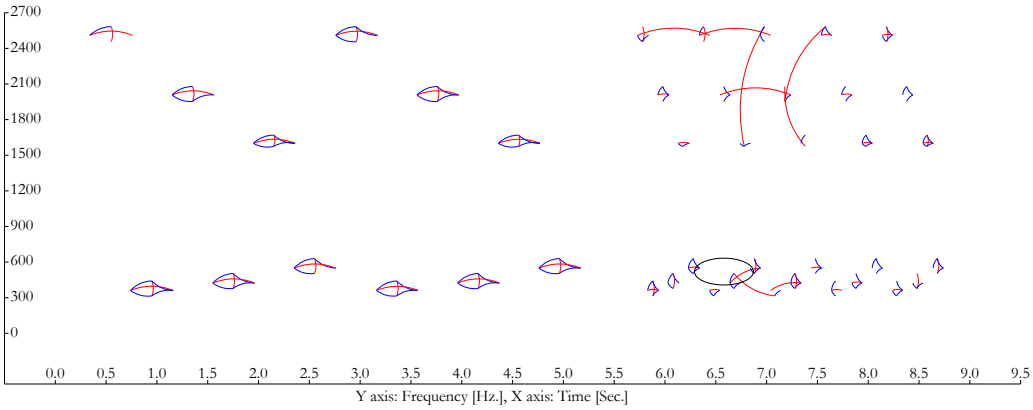

Figure 8-18: context test, track 1. Display of existing bonds only

The next two figures show the same information, but at the end of the run. This time, many more bonds have lost fights (Figure 8-19), while the existing bonds seem to converge to the intended grouping (Figure 8-20).

Frame 800: a newly proposed bond fights and loses against already built bonds

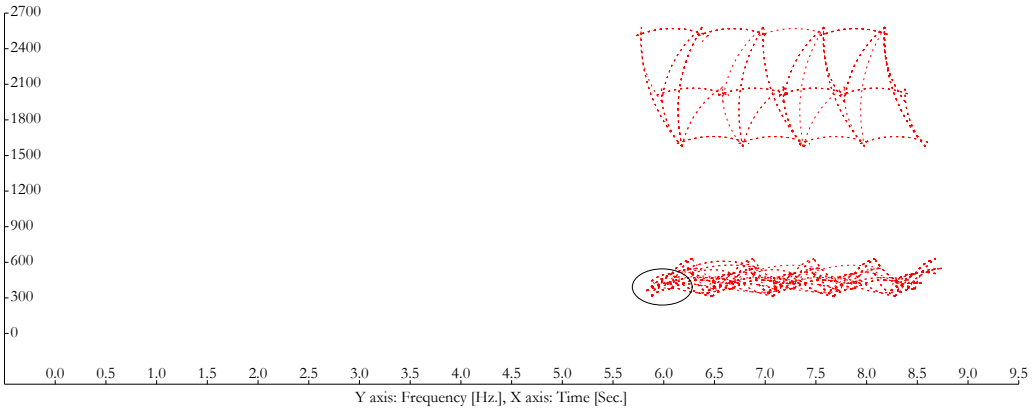

Figure 8-19: context test, track 1. Display of dead bonds only, end of run

Frame 800: a newly proposed bond fights and loses against already built bonds

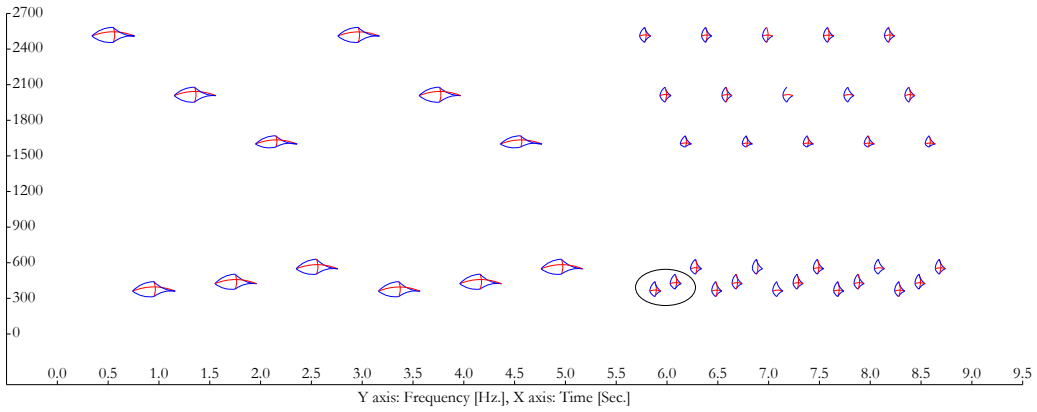

Figure 8-20: context test, track 1. Display of winning bonds only, end of run

I conclude this section with a detailed view of the right lower region of the example landscape showing only existing bonds (which either had no fights yet, or had them and won) for clarity. The four figures are arranged sequentially according to the original order of activity.

Initially (Figure 8-21), only a few bonds are built, some better than others.

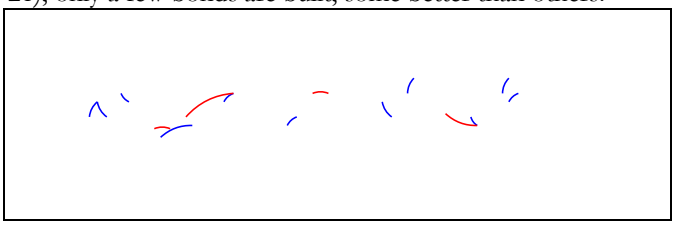

Figure 8-21: context test, track 1, lower right region, emergent bond building

Next, as more bonds are built, more of them are 'wrong'. Some of the non-ideal bonds from above were replaced by what seemed to be better ones at the time (Figure 8-22).

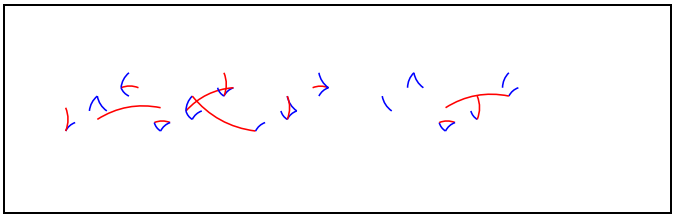

Figure 8-22: context test, track 1, lower right region, emergent bond building

As the run progresses (Figure 8-23), more mutual supporting bunches emerge, which will be more successful at fighting conflicting bonds. Some of the 'wrong' bonds (in this context) have not yet lost a fight. There is nothing wrong with that, since no strong support have been found at to that point to suggest that these bonds are wrong.

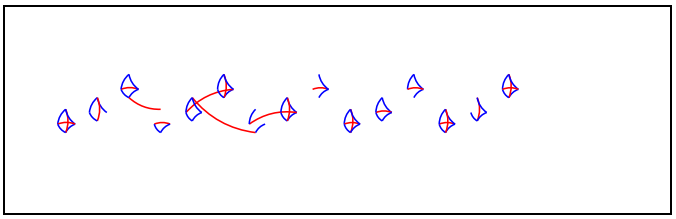

Figure 8-23: context test, track 1, lower right region, emergent bond building

Close to the end of run, only bonds with strong mutual support survived, see

Figure 8-24. All existing bonds seem to us to be correct just because they do not conflict with others. This is what makes them seem 'correct'. Notice that some 'missing' bonds have not yet been proposed. We see that from context as well, but the current implementation does not deal with such higher-level capabilities.

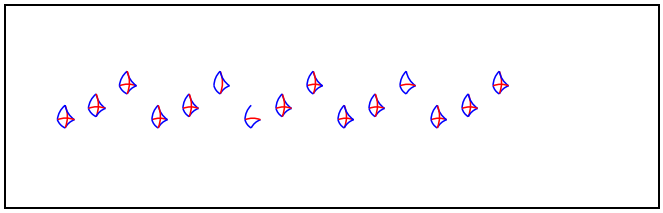

Figure 8-24: context test, track 1, lower right region, emergent bond building

8.1.3 Proximity

For the preliminary phase of the proposed model, I chose to implement as a test auditory scene analysis grouping pressure the proximity scout. This grouping pressure shows in psychoacoustic experiments the tendency to favour the grouping of those tones that are close together by both time and frequency. Many unknowns exist regarding the interpretation of such experiments. The following is a list of some of these unknowns. Is there a single temporal-spectral proximity agent, or two (or more) separate pressures (perhaps competing on frequency and time proximity)? Do these grouping pressures work on whole tones, or on distinct primitive derivative cues (such as positive first derivatives, or even many different types of cues in parallel)? How does competition take place among conflicting pressures? In any case, enough evidence exists to show that competition occurs between time and frequency distances within the proximity pressure, and that the proximity grouping pressure interacts (depending on the context by either supporting or resisting) other grouping pressures. Experiments were designed to eliminate the working of all other grouping pressures (such as harmonicity, common fate, etc.) and investigate the proximity grouping in isolation. In this section, I test the behaviour of the proximity scout on some of these sound fragments, starting with the sound fragment from track 3 of Bregman's demonstration CD. Due the length of the sound fragment, it was split into 6 sections. Sections 1 to 3 contain tones whose frequencies have a high-low-high pattern. While the frequency separation remains constant, the time between the tones gradually diminishes, and the tones become shorter. Humans tend to perceive the fragment initially as if it was played by a single source of sound whose frequency changed from high to low and back again. With time, though, as the duration of tones and the gaps between them become smaller, there is a shift in perception, and we start hearing two separate streams. One stream consists only of the higher tones, while the other only of the lower ones. Rhythmically, the perceived pattern changes from a galloping rhythm (high-lo-high…high-lo-high…) into two streams with isochronous tones (high…high…high… and low……low……low…). Since the only change in the fragment is the time scale, the idea is that the proximity pressure initially favours to group tones bridging the frequency gaps, while it later shifts into favouring the time gaps as they become shorter (see Bregman 1990, p. 17-21, 48-73). The importance of this experiment is in the existence of the shift itself, rather than the actual point at which it occurs, since that point can vary among subjects and each subject may experience slight differences each time he or she listens to the fragment. Figure 8-25, Figure 8-26 and Figure 8-27 show the spectrograms of sections 1, 2, and 3 respectively. Notice how the time scale changes over time, while all other parameters remain constant.

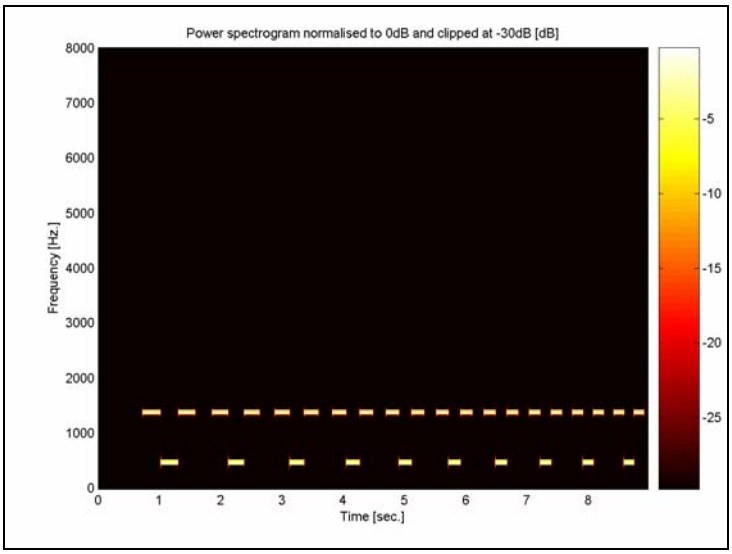

Figure 8-25: Proxímity test, track 3a

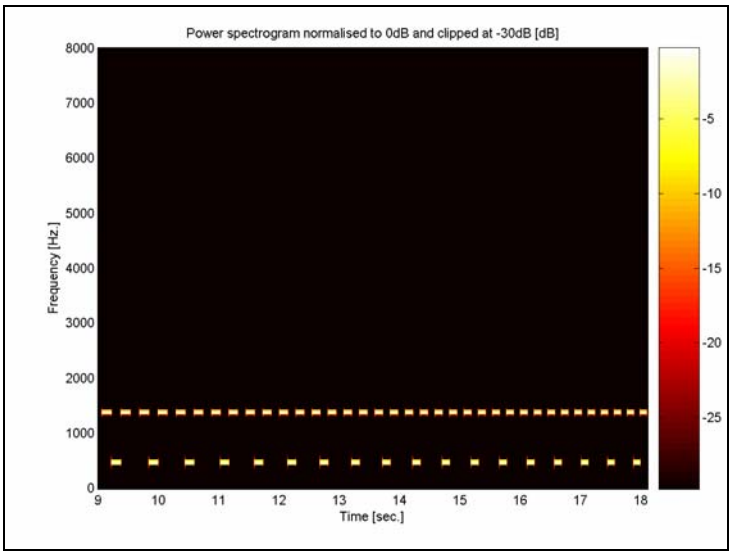

Figure 8-26: Proxímity test, track 3b

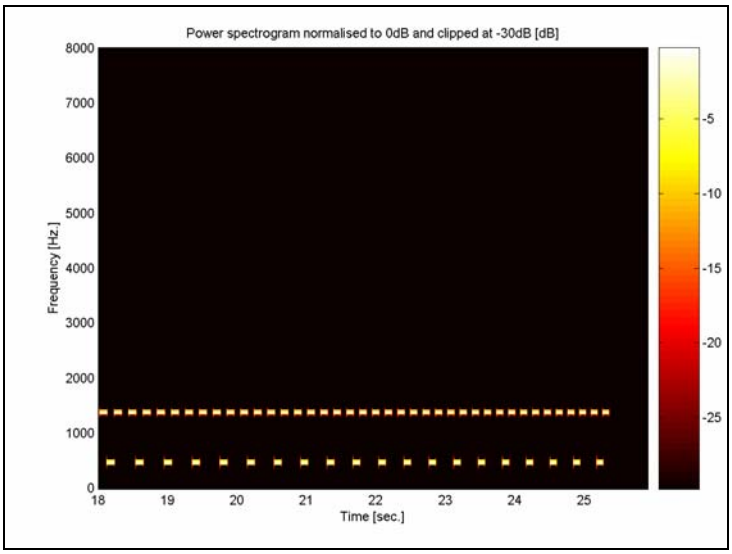

Figure 8-27: Proximity test, track 3c

Sections 3 to 6 repeat the same pattern of sections 1 to 3 with the only difference being the constant frequency gap. In the first three sections, it was 17.8 semitones, while in the last three it is one semitone.

*Figure 8-28**Figure 8-29*Figure 8-30 depict the spectrograms of sections 4, 5 and 6.

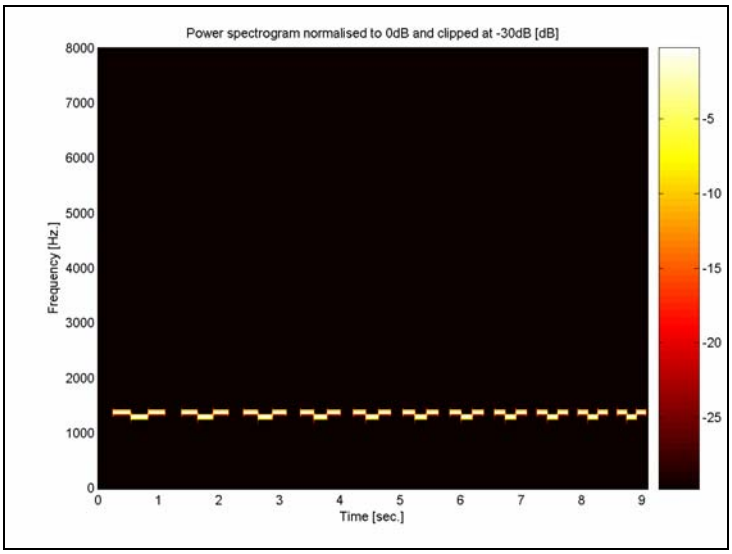

Figure 8-28: Proximity test, track 3d

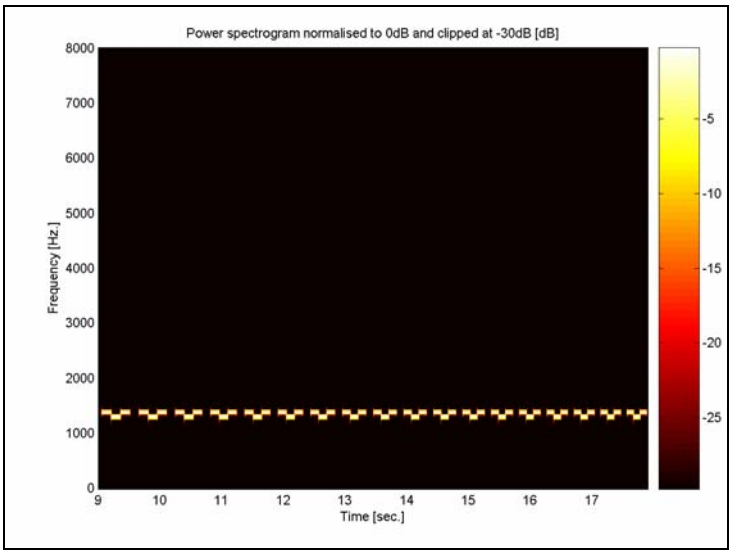

Figure 8-29: Proximity test, track 3e

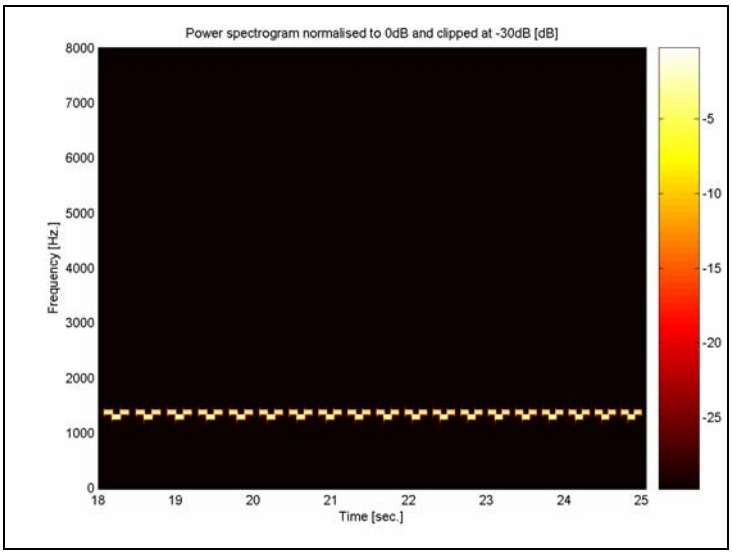

Figure 8-30: Proximity test, track 3f

The proximity scout measures proximity in the following way:

$$proximity = (t_2 - t_1)^2 + (0.22 \log_2 \frac{f_2}{f_1})^2$$

This gives a score that is interpreted by the proximity scouts, and is used to settle fights between conflicting proximity bonds. The log takes care of the (near) logarithmic frequency used by humans. The weighting constant was found by trial and error when working with this example. Other sound fragments were then used to confirm that such a formula works. So far it does, which might suggest that at least in principle such approximation works. Notice, however, that this measure is not global, as it is only used by local scouts in local fights. The interaction of the

proximity scout with all other activities of *The Ear's Mind* will affect the results much more than the effects of small parameter tuning of the proximity formula. For instance, if the proximity scout will only be allowed to bond positive first-derivative cues which already function as onsets of existing bunches, than no cues will be connected by proximity bonds, which have not yet been titled as onsets. Likewise, cues that have been bonded by proximity should not be allowed to reside within a single bunch unless the bond is destroyed in a fight. The idea that proximity bonds should fight each other seems to solve the problem of finding the ideal proximity measure. Instead, since fights are allowed, proximity measurements become relative to other proximity bonds. In addition, no two proximity bonds are allowed to either start or end from (or on) the same cue. How this works can be seen in the following example, using track 3, sections 1 to 6. In this section, proximity tests were done in isolation with no interaction with other scouts or activities. Moreover, the proximity scouts were only allowed to 'see' positive temporal derivatives. Figure 8-31 shows the input list of positive temporal derivative cues as *The Ear's Mind* received from the preprocessor.

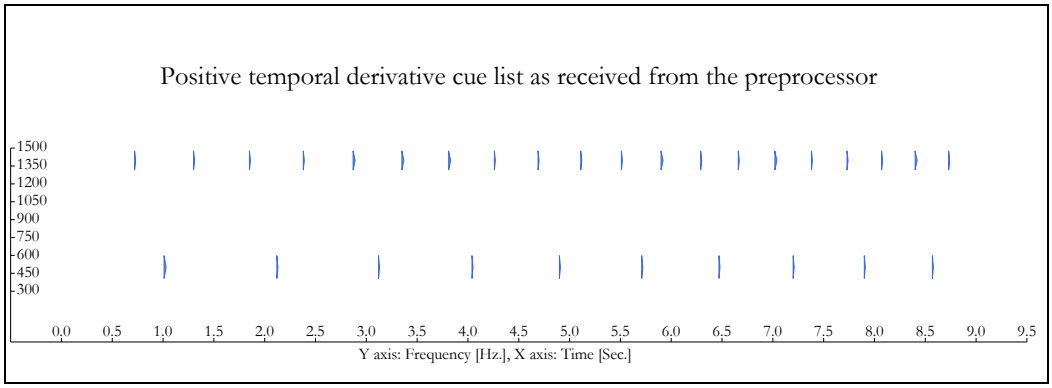

Figure 8-31: Proximity test, track 3a, starting point

Figure 8-32 below depicts the first proximity bond to be built.

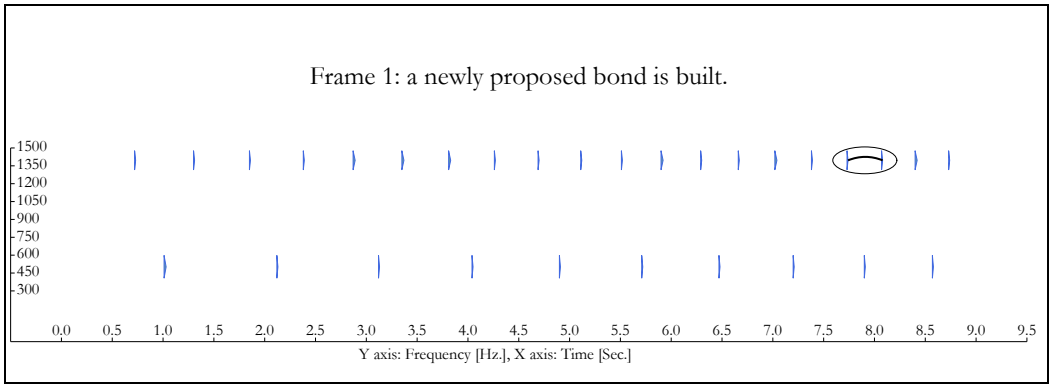

Figure 8-32: Proximity test, track 3a

Some frames later (see Figure 8-33), another scout tries to build a proximity bond which ends at the same cue of an existing bond. This conflict has to be settled by a fight, which ends by the existing bond winning, and the losing newly proposed bond removed and rendered in dashed spline.

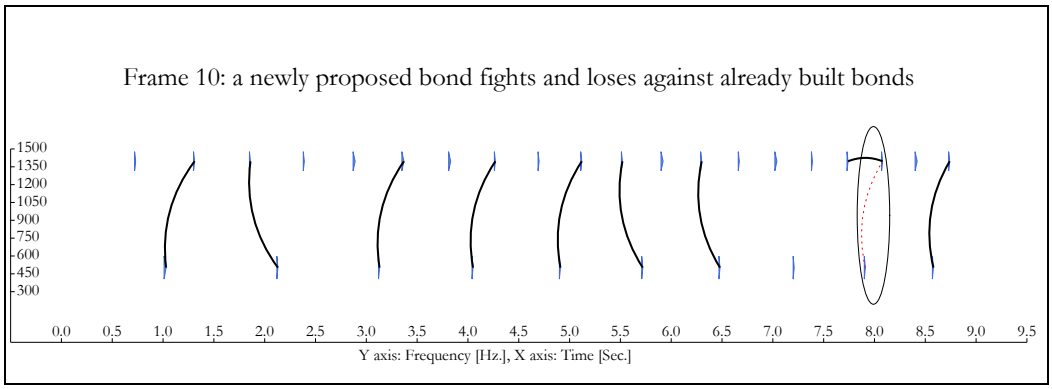

Figure 8-33: Proximity test, track 3a

Skipping several frames ahead (see Figure 8-34), we see that more bonds have been built, and together with some losing bonds, a pattern emerges. At the beginning (left) a line (stream) emerges that bonds alternating high and low cues, while later on, (right) more and more bonds following the pattern of the left side lose to horizontal bonds between successive high frequency cues. This is the working of the local activity of the scouts. Context (in the form of the presence of other cues not known to the local scout proposing the bond) affects the future of the proposed bond. The control mechanism for achieving that is no different from any other interaction in *The Ear's Mind* so that any context may influence any activity in any direction.

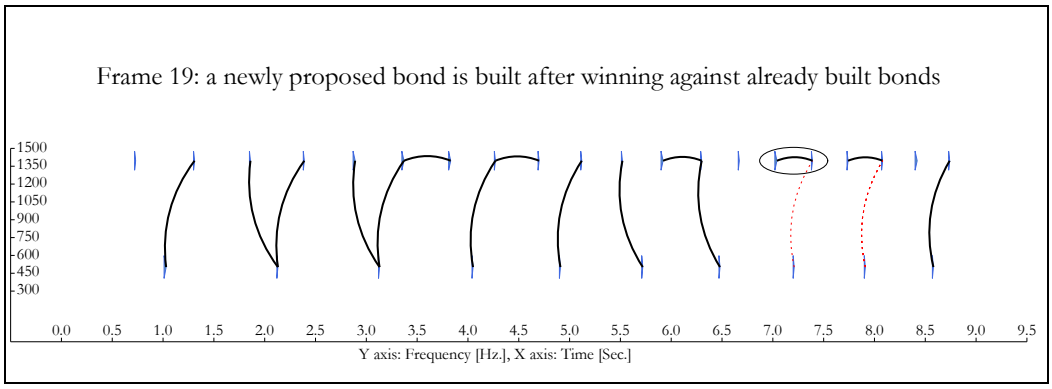

Figure 8-34: Proximity test, track 3a

Finally, Figure 8-35 below shows a stable outcome of section 1. The right side could be interpreted as a transition zone where the one stream interpretation is no longer valid, though a two-stream view has not yet been built. Since I have not expected this result, I was surprised to see it emerge, and feel that it certainly does not contradict the perception of human subjects. Moreover, it may strengthen psychoacoustic evidence since such regions of weakness of structure (where the structure gives way to pressures of other mechanisms) can certainly be observed in many cases. *The Ear's Mind* does not have any mechanism or knowledge of how many streams we expect to have in this, or any other fragment. It does not even have a clue of what streams are.

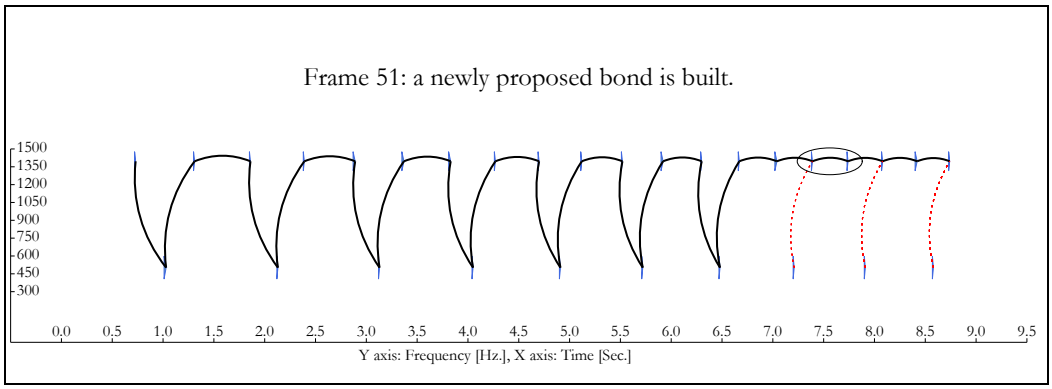

Figure 8-35: Proximity test, track 3a, end of run

Section 2 results are given in Figure 8-36 below. The same semi-stream interpretation carries on throughout as started at the end of section 1. The results of section 3 are given in Figure 8-37.

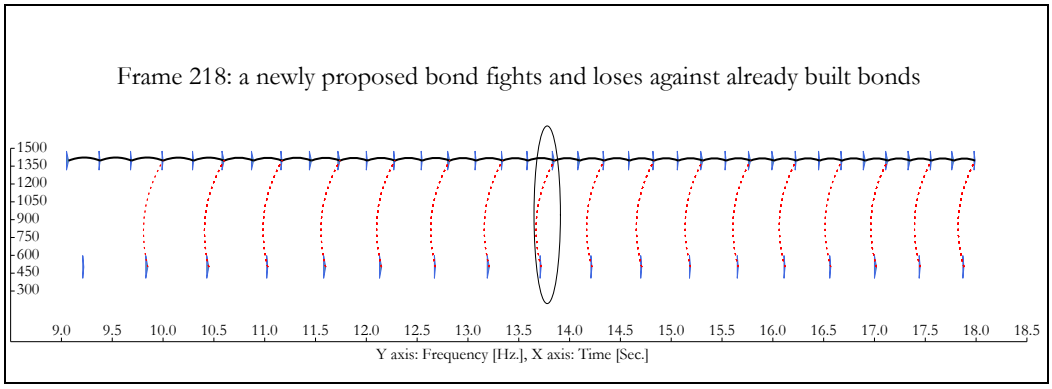

Figure 8-36: Proximity test, track 3b, end of run

In the 3rd section (Figure 8-37) we see the formation of a second stream, which takes over from the semi-stream interpretation of section 2. by the use of the proximity scout within an emergent framework an interpretation emerged which transforms from a one stream interpretation through a semi- (or weak) stream to a strong two stream interpretation. Notice that scouts sensitive to isochronous tones, or rhythm scouts, if released on cues attached by streams will produce the galloping versus isochronous rhythm patterns that match the perception of the listener. I have started this work by stating that what we hear is our perception of things, not what really is out there. This change in rhythm has nothing really to do with the input signal in itself. It has to do with our interpretation of it. These are, of course, preliminary results based only on proximity pressures in isolation. *The Ear's Mind*, as I have repeated throughout this work needs the interaction of many types of features and scouts to produce complex behaviour and approach the auditory scene analysis goal.

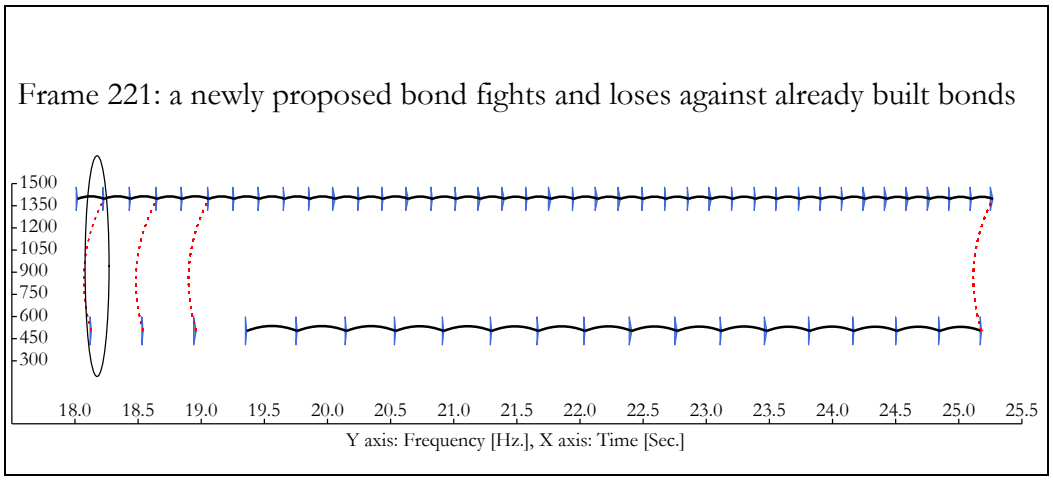

Figure 8-37: Proximity test, track 3c, end of run

When the frequency gap is reduced to one semitone, things become different. Although *The Ear's Mind* starts with a single stream in section 4 (Figure 8-38), the same interpretation is strong enough to persist throughout the fragment. See results of section 5 and 6 in Figure 8-39Figure 8-40 respectively.

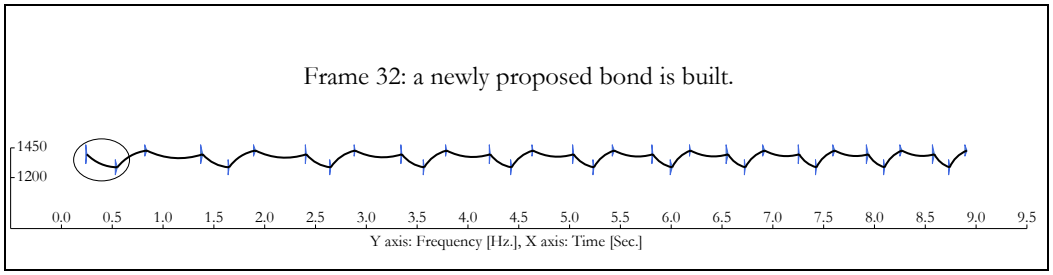

Figure 8-38: Proximity test, track 3d, end of run

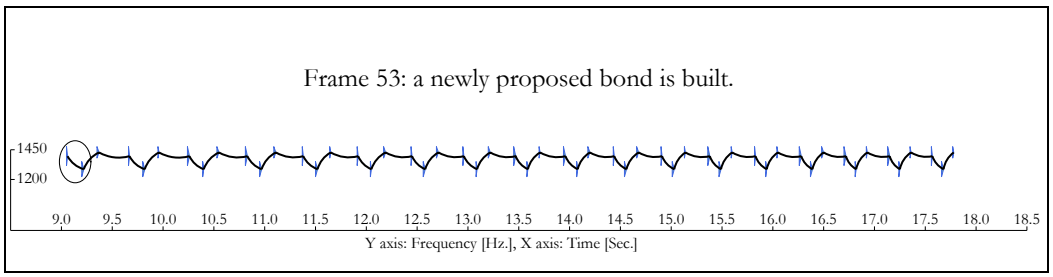

Figure 8-39: Proximity test, track 3e, end of run

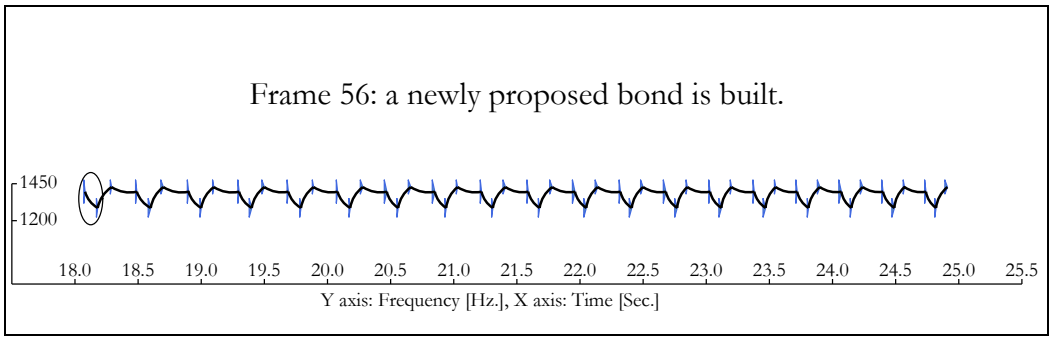

Figure 8-40: Proximity test, track 3f, end of run

From this perspective, note the use of the semitone (as well as the whole tone) in music for creating the illusion of movement, or accentuating a note without breaking up the melody line into separate streams (e.g. the trill).

In the next test, since I have tuned the proximity formula above using track 3, it is interesting to note the behaviour of the proximity scout on other tracks in isolation. Here, I include track 1, which was used in the previous section to test the workings of bunches and families. The procedure is the same as the above tests on track 3. In Figure 8-41 on the left, we see the one stream (weak in some places, the cause of which should be investigated) interpretation that matches listener's perception. The right side matches the perception of human subjects. We start hearing two separate streams – two melodies in contrary motion as if played by two instruments.

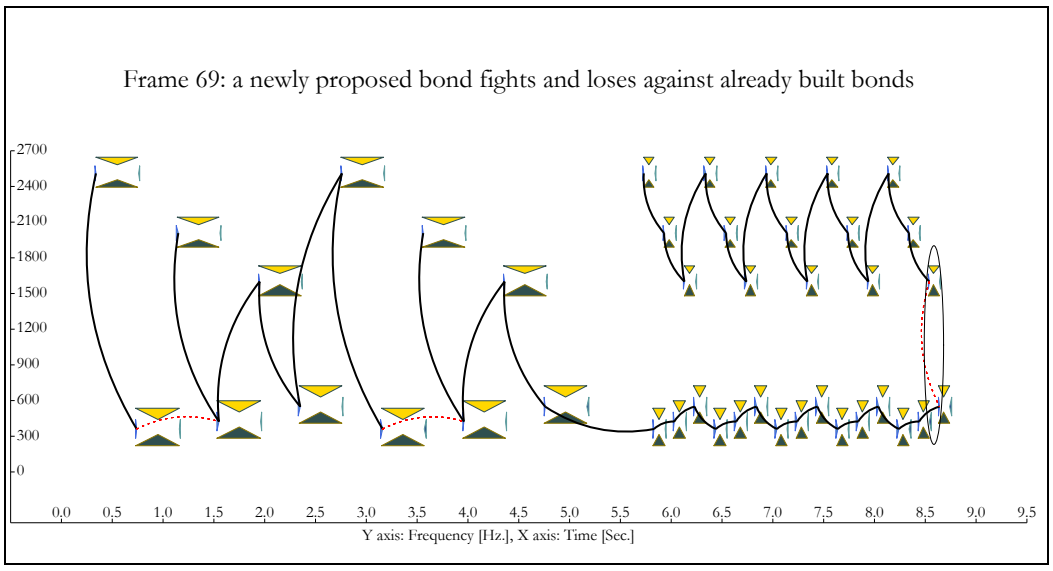

Figure 8-41: Proximity test, track 1, end of run

The same experiment carried out on track 2 from the demonstration CD produces the following results (see Figure 8-42).

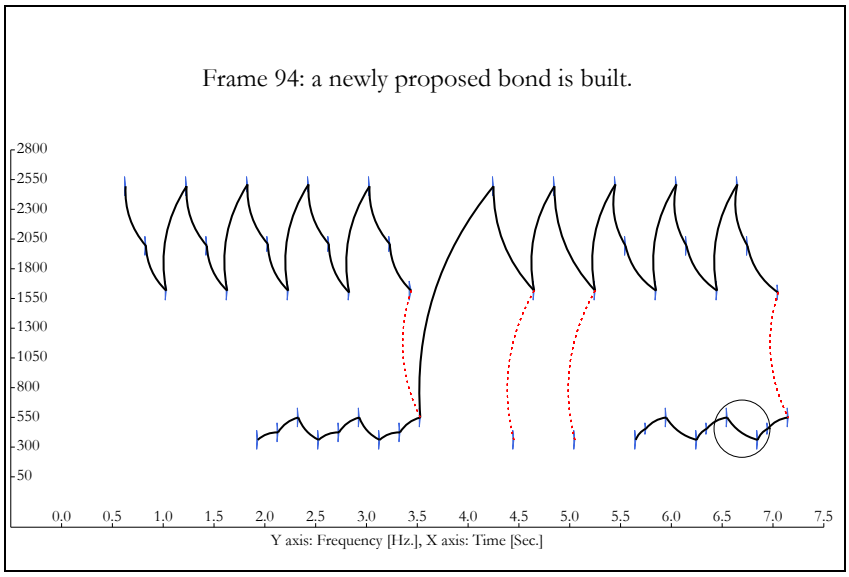

Figure 8-42: Proximity test, track 2, end of run

In this experiment, the listener is asked to attempt to 'hear' the preceding stream (the so-called standard) while listening to the following notes (the full cycle). In the first case, three tones close in frequency are played, and when the low tones are added to the mix, there is no problem for subjects to hear the same notes separately from the low one, and keep 'tracking' them. As can be seen from the above figure, *The Ear's Mind* has no difficulty with this either, because the three high notes form a separate stream from the low ones. However, in the second half the presented standard is formed from both high and low notes. Though it is possible to follow this melody in isolation, as soon as the full cycle starts, it becomes extremely difficult to do, since the notes split once more to the two streams of high and low frequencies. *The Ear's Mind* results match this behaviour as well. In the figure, it can be seen that only a weak stream is formed in the second standard, while the second full cycle produces the same two-stream interpretation as the first one.

8.2 Proposed future work

With the implementation of *The Ear's Mind*'s skeleton, a minimal set of scouts and basic functionality, the desired emergent behaviour is showing the potential of this approach. One of the theoretical principles of this model is that the ability to approach human auditory primitive perception will only become reality if a redundancy of evidence from different features extracted from the input fragments, together with active pressures are allowed to take part in the self-organising soup on the way to perception. Now that the basic concept is verified, it is proposed that new modules will be implemented and research will carry on developing the theory of the model. The following points will be targeted in the next phase of *The Ear's Mind*:

- ***Preprocessing.*** Implementing new reprocessing algorithms for the extraction of new features (energy levels, micro-modulation, etc.) together with a more robust implementation of existing features allowing the creation of multilevel cues and thresholds. Allowing the definition of angled cues.

- ***Scouts*** for the new features, with multilevel capabilities etc. Defining new scouts for harmonicity, isochronous and rhythmic relations etc. Additionally, mechanisms should be investigated for the implementation of competition between scouts, and the working of bunches and families while keeping all activities local.

- ***New Tests*** for interaction and integration of the various grouping pressures. Tests for investigating strategies of using context for grouping sensibly.

- ***Hierarchic levels*** of abstraction including groups, groups of groups and their ability to put both bottom-up and top-down pressures on each other.

- ***Programming*** methods should be investigated for deducing the right local rules that bring about the desired emergent behaviour.

- ***Real-world*** sound fragments should be used at that point with some functionality. Work would be done targeting separation of sound sources, though no recognition of sources will result at that stage.

- ***Domains*** other than the auditory domain should be investigated for possible tailoring of the proposed model and the implementation of The Eye's Mind, etc.

8.3 Conclusions

I have proposed *The Ear's Mind* model as a novel emergent, self-organising computer model for auditory scene analysis. To come up with a design, which is both biologically plausible, and extendable, I have reviewed related fields of research and laid out requirements and constraints for the model. Additionally, I have conducted an extended survey of existing CASA models and critically reviewed them in relation to the proposed model. Consequently, the proposed model is, in my view, both theoretically founded and innovative. Basing *The Ear's Mind* on the architecture of the Copycat model, new modules for preprocessing and display as well as domain specific modules for auditory perception were designed. The model consists of an open architecture allowing the addition of new features, pressures and interaction methods, making it possible to define a first implementation phase for preliminary tests of the theory and to extend the model's functionality later on.

Following the design phase, the model was implemented as a software prototype, and was used for testing its ASA capabilities. Such tests were carried out using sound fragments from psychoacoustic experiments and results were compared to those of human subjects. Preliminary results showed that the model behaves as expected, and even produces some unexpected emergent behaviour, which matches that of human subjects in experiments of isolated proximity grouping pressures. These results favour the conclusion that the implemented model is working as expected and forms a promising foundation for further research. Consequently, a plan has been devised for the second phase of implementation. It is hoped that this work will add to the growing amount of literature on the subjects of sensory perception and the cognitive sciences. If this work will help others in gaining insight on these fascinating fields, I consider it a success.

9 Bibliography

1. **Barr A. et al,** editors (1989), *The handbook of Artificial Intelligence, volume IV,* Addison-Wesley Publishing company, Inc. (*chapter XVI: Blackboard Systems)*

2. **Bilsen F. A.** (1985), *Akoestische Perceptie,* Vakgroep Akoestiek, Technische Natuurkunde, TU Delft

3. **Bregman A. S.** (1990), *Auditory scene analysis, the perceptual organisation of sound*, 2nd paperback ed. 1999, MIT Press

4. **Camazine, S. et al,** editors, (2001), *Self-organization in biological systems,* 2nd paperback ed. 2003, Princeton university press

5. **Cook P. R.** (1999), *Music, Cognition, and Computerized sound: An introduction to Psychoacoustics*, MIT Press

6. **Cooke M. et al,** editors, (1993), *Visual Representations of Speech signals,* JOHN WILEY & SONS

7. **Cooke M.,** (1993), *Modelling Auditory Processing And Organisation,* Cambridge University Press

8. **Dallos P. et al,** editors (1996), *The Cochlea*, Springer – Verlag

9. **Dennett D. C.** (1998), *Brainchildren, Essays on designing minds*, MIT Press

10. **Edelman G.,** (1992), *Bright Air, Brilliant fire, On the Matter of the Mind,* PENGUIN BOOKS

11. **Ehret G. & Romand R.,** editors (1997), *The Central Auditory System,* Oxford University Press, Inc.

12. **Ellis, D. P. W.,** (1996), *Prediction-driven computational auditory scene analysis, PhD Thesis, MIT*

13. **Eysenck M. W. & Keane M. T.** (1995), *Cognitive Psychology, a student's handbook*, *3rd ed.*, Psychology Press

14. **Fishbach, A.,** (1994), *Primary Segmentation of Auditory Scenes,* 1051-4651/94, IEEE

15. **Fishbach, A.,** *Auditory Scenes Analysis: Primary Segmentation and Feature Estimation*

16. **Franz Inc.** (1988), *Common Lisp, The reference,* Addison-Wesley Publishing Company, Inc.

17. **Gardner H.** (1987), *The Mind's new science, A history of the cognitive revolution*, Basic Books

18. **Goldstein E. B.** (2002), *Sensation and Perception*, 6th ed. Wadsworth

19. **Graham, P,** (1994), *On Lisp, Advanced techniques for Common Lisp, Prentice Hall*

20. **Haus G.**, editor (1993), *Music Processing*, Oxford University Press

21. **Helmholtz H.** (1885), (reprinted 1954), *On the sensations of tone*, Dover Publications, Inc.

22. **Hofstadter D. R. et al** (1979), *Gödel, Escher, Bach: An Eternal Golden Braid, A metaphorical fugue on minds and machines in the spirit of Lewis carol,* Basic Books

23. **Hofstadter D. R. et al** (1995), *Fluid concepts and creative analogies, computer models of the fundamental mechanisms of thought*, Basic Books

24. **Holland J. H.** (1998), *Emergency, from Chaos to Order*, Oxford University Press

25. **Kinsler L. E. et al** (1982), *Fundamentals of acoustics, third ed.*, John Wiley and sons

26. **Lerdahl, F., R.S. Jackendoff** (1983), *A Generative Theory of Tonal Music*, Cambridge MA: MIT Press. Cambridge: Cambridge University Press

27. **Lewis R.,** (1999), *Complexity, Life at the edge of chaos,* the university of Chicago Press

28. **Marais E. N.** (1937), *The soul of the White Ant, Methuen & Co. Ltd. London*

29. **Marr, D.,** (1982), *Vision,* New York: W. H. Freeman

30. **Marshall B. & Hofstadter D. R.**, (1996), *Beyond Copycat: Incorporating Self-Watching into a Computer Model of High-Level Perception and Analogy-Making,* paper of the Center for Research on Concepts and Cognition & department of Computer Science, Indiana University

31. **Mathews M. V. & Pierce J. R.**, editors (1989), *Current directions in computer music research*, MIT Press

32. **Matthews G. G.** (1998), *Neurobiology, Molecules, Cells, and Systems*, Blackwell science

33. **Miranda E. R.,** editor, (2000), *Reading in Music and Artificial intelligence,* Harwood academic publishers

34. **Mitchell M.,** (1993), *Analogy-making as Perception, A computer model,* MIT Press

35. **Moore B. C. J.,** editor (1995), *Hearing: Handbook of Perception and Cognition*, *2nd ed.* Academic press

36. **Moore B. C. J.,** (1997), *An Introduction to the Psychology of Hearing, fourth ed.,* Academic press

37. **Nakatani T. et al,** *Auditory Stream segregation in Auditory Scene analysis with a Multi-agent System (research paper)*

38. **Norvig P.** (1992), *Paradigms of Artificial intelligence programming: case studies in common Lisp*, Morgan Kaufmann publishers, Inc.

39. **Prigogine I.** (1984), *Order Out of Chaos, man's new dialogue with nature,* Bantam books

40. **Quatieri Th. F.,** (2002), *Discrete Time Speech Signal Processing principles and practice,* Prentice Hall, Inc

41. **Ramachandram R. P. & Mammone R.**, editors (1995), *Modern Methods of Speech Processing,* Kluwer Academic Publishers

42. **Rosenthal, D. F. & Okuno H. G,** editors (1998), *Computational Auditory Scene Analysis,* Lawrence Erlbaum associates, Inc., Publishers

43. **Russell, S. & Norvig P.** (1995), *Artificial intelligence, a modern approach*, Prentice Hall

44. **Sacks, O** (1987), *The Man Who Mistook His Wife for a Hat*

45. **Sacks, O** (1995), *An Anthropologist on Mars*

46. **Schwanauer S. M. & Levitt D. A.,** editors (1993), *Machine models of music*, MIT Press

47. **Todd P. M. & Gareth L. D.**, editors (1991), *Music and connectionism*, MIT Press

48. **Todd P. M. & Griffith N.**, editors (1999), *Musical Networks*, MIT Press

49. **Waibel A. & Lee K.,** editors (1990), *Readings in Speech Recognition,* Morgan Kaufmann Publishers, Inc. (*chapter 5: Knowledge-Based Approaches)*

50. **Welch B. B. et al,** (2003), *Practical Programming in Tcl/Tk,* Prentice Hall PTR

51. **Yost W. A. et al,** editors (1993), *Human Psychophysics,* Springer – Verlag