

A hybrid approach to symbol grounding

Bridging the semantic gap between connectionist and symbolic systems in an automated fashion

Thomas Markus

First supervisor and reviewer:

dr. ir. Jan Broersen

Master's thesis
Cognitive Artificial Intelligence
Utrecht University
30 ECTS

Second supervisor and reviewer:

dr. Thomas Müller

Third reviewer:

prof. dr. John-Jules Meyer

April 6, 2009

Abstract

A common design practice in hybrid agent architectures is to create a static mapping from connectionist low level processing to high level symbolic states. A dynamic mapping, as advocated in this master's thesis, yields considerable advantages from a software engineering point of view allowing automatic exploitation of newly added symbolic information or extra sensor data without manual intervention. A closely related problem exists in a philosophical context between narrow and wide scope semantics in Conceptual Role Semantics and concept learning. Solving the engineering problem will shed some light on the philosophical problem. A framework is proposed that uses Fuzzy Adaptive Resonance Theory (ART-2) for creating perceptual categories and a genetic algorithm for learning the mapping from those to symbols. Little is assumed about both the symbolic system and the sensors. This makes the framework applicable to a wide range of tasks. Results illustrating some common situations show, that the approach works in various environments and that the proposed extensions to the genetic algorithm support incremental learning.

Acknowledgements

I would first like to thank my thesis advisor Jan Broersen for his thought provoking remarks and fun and relaxing discussions. Thomas Müller was invaluable for restructuring large parts of this thesis and thorough advice about the essentials. John-Jules Meyer was indispensable for highlighting some previously unattended points regarding corners that were cut too short.

Contents

1	Introduction	1
1.1	Neuro-symbolic computation	1
1.2	Agent engineering perspective	2
1.3	Philosophical perspective	3
1.4	Research questions	6
2	Toy problem	9
2.1	Parallels and components	9
2.2	Recognizing arithmetic	10
3	Conceptual nativism & hybrid systems	13
3.1	Symbolic limitations	13
3.2	Categorization	15
3.3	From symbols to concepts	16
3.4	Extreme nativism	17
3.5	Differentiating concept competence	20
4	Machine learning	23
4.1	Introduction	23
4.2	Types of learning	23
4.3	Genetic algorithms	26
4.4	Adaptive Resonance Theory	29
5	The framework	33
5.1	Overview	33
5.2	Symbolic system	34
5.3	Sensors	35
5.4	Horizon	35
5.5	Genetic algorithm	36
5.6	Protective mutation	38
5.7	Resource sharing	38
5.8	Runtime	40
6	Results	43
6.1	Fixed horizon	43
6.2	Dynamic horizon	46
6.3	Performance	47
6.4	Protective mutation	48
6.5	Adding a sensor	49
6.6	Learning a new symbol	52

7 Conclusion & Discussion	53
7.1 Conclusion	53
7.2 Discussion	55
7.3 Further research	55
Bibliography	58

Chapter 1

Introduction

1.1 Neuro-symbolic computation

Symbolic and sub-symbolic computation are largely viewed as complementary approaches to artificial intelligence. Symbolic systems are often deployed in a setting where the information is easily formalized and isn't noisy. On the other end of the spectrum are sub-symbolic (connectionist) systems which have successfully solved diverse tasks such as temporal pattern recognition in video, handwriting recognition and games. Some types of connectionist systems are inspired by the brain's neural structure, leading to the name of Neuro-symbolic computation. This is the research area that tries to integrate connectionist and symbolic systems.

The application domains are relatively disjoint, but introducing more structure in connectionist systems and greater flexibility in automated learning in symbolic systems can yield remarkable improvements [29]. These improvements call for an increased use of so called hybrid systems that exploit both intrinsic strengths. I.e. some aspects of a problem may be analog and noisy, but eventually lead to high level states with complex functional interrelations. These relations could then be captured by some symbolic system with some rigorous formal theory behind it.

Lots of questions immediately come up such as:

- What kind of information needs to be exchanged between these two systems in a single agent?
- Can one type of system be reduced to the other one by means of some automatic translation?
- How can one interpret a neural net as a symbolic system in a generic automated way?

The field of neuro-symbolic computation covers the full spectrum of these types of questions with very different research directions. Some of them are complementary or even conflicting:

- It is possible to generate connectionist systems that fulfill the role of symbolic systems, but with a greater resistance to noise. [1]
- The reverse approach where rules are extracted from “single neural network”-equivalent systems or an ensemble of networks is very active as well. [19]
- Formal models, defining the models of interaction between symbolic and connectionist components in a single system, have been researched as a separate subject [42, 14]. Today it is more common to integrate this into a larger agent architecture.

In this new field many challenges remain [29] including the *semantic barrier* (how to go from intrinsically meaningless sensor outputs to meaningful symbols) between the two when you want to combine them. This thesis will focus on bringing down the semantic barrier, which can be tackled starting from two very different perspectives, namely: *Agent Engineering* and *Philosophy of Artificial Intelligence*.

1.2 Agent engineering perspective

The agent engineering perspective shows us a possibility for automating a common design practice of current hybrid systems. Currently the interpretation of the output of a connectionist system, when further used in some kind of symbolic system, is often hardcoded. I.e. the designer of the system determines the interpretation of connectionist sensor data and manually defines which symbols should be associated with those outputs. This is undesirable though, for the following reasons:

1. The connectionist system needs to be trained manually.
2. The connectionist system does not profit from increased information in the symbolic system.
3. The connectionist system needs to be retrained when the environment changes and the connectionist-symbolic-link then needs to be manually redefined.

To prevent the drawbacks just mentioned, one wants to learn the interpretation (the translation of output patterns to some symbolic expression) in an automated fashion. This offers advantages from the software engineering point of view, such as: reusability of existing intelligent systems and automatic selection of task specific machine learning algorithms. The proposed method is more of a framework for building on the intrinsic strengths of the two classes of systems than an attempt at translating the one into the other. Furthermore, the proposed system will try to make this relation between sensor outputs and symbols adaptive, automatic and will allow for multiple ambiguous interpretations to co-exist. The basic concept of the division of labour between the two types of systems and their interactions is applicable to a wide range of tasks and profits from the advantages that both types of systems intrinsically have.

The agent's task is briefly defined as follows: For any random set of inputs I : find an approximately correct mapping for all $i \in I$ that maximizes some evaluative feedback value obtained by evaluating the resulting expression in some symbolic system.

This task is interesting because it introduces a number of problems, or to put it more optimistically, *challenges* which will have to be met:

1. The complete search space of all mappings is n^m (assuming no optimizations whatsoever), where m is the number of computational percepts (templates) and n is the number of symbols in the symbolic system.
2. Many local minima exist in mapping the computational percepts to symbols.
3. Evaluation is only available for a complete mapping, obscuring the actual partial mapping that is in error. The correct mapping (global optimum) can only be discovered by comparing feedback for the same mapping on different expressions.
4. Connectionist outputs have no intrinsic meaning nor do symbols. What happens when symbols don't have a neatly fitting connectionist counterpart? And how do analog connectionist values connect to discrete representations without severe information degradation and loss of noise resistance and generality?

These are only a few examples of the challenges that will need to be solved in the agent engineering task and many more remain. Connectionist sensor output will have an effect on the symbolic system and changes in the symbolic system will be reflected in the use of the connectionist sensors data and its categorization.

What the system is meant to do is perhaps most clearly summarized by [32]:

A common element in these models is that they sort continuous sensory input represented as feature vectors into discrete categories that are associated with labels according to linguistic convention. Categorization may be modeled through either generative or discriminative methods. [32, p.389]

This is in contrast to many approaches which mention symbol grounding (such as the formation of an "audio-visual lexicon"), but seem to ignore the difference between categorization and identification. This is the engineering challenge in a nutshell.

1.3 Philosophical perspective

Now turning to the philosophical perspective which perhaps isn't as clear at first sight, we see a different challenge that has strong ties to the study of grounding concepts in symbolic systems.

Searle's infamous 'Chinese Room'-argument [34] claimed that symbol systems have some fundamental limitations preventing them from exhibiting true cognitive behaviour. This is in large part due to the use of ungrounded symbols in such systems. The 'Chinese Room'-argument will be more extensively covered in section 3.1. Harnad [11] however argued that a system with sensors and connectionist systems classifying that raw sensor data would not be subject to the 'Chinese Room'-argument when using the right approach.

This approach would amount to grounding symbols in the environment (or more precisely low level 'perceptions'¹).

Luc Steels is quite confident that current advances have made the symbol-grounding problem obsolete [36]. Steels however constrains the domain of grounded symbols considerably to justify such a bold statement:

In some cases, there is a method that constrains the use of a symbol for the objects with which it is associated. The method could, for example, be a classifier a perceptual/pattern recognition process that operates over sensorimotor data to decide whether the object 'fits' with the concept. If such an effective method is available, then we call the symbol grounded. There are a lot of symbols which are not about the real world but about abstractions of various sorts, like the word 'serendipity', or about cultural meanings, like the word 'holy water', and so they will never be grounded through perceptual processes. [35, p.223]

The symbol grounding problem from the perspective of Steels concerns whether it is possible "to ever conceive of an artificial system that is able to invent and use grounded symbols in its sensorimotor interactions with the world and others." [35] (page 240). Steels gives numerous examples of agents that invent and calibrate new languages that emerge from self organizing classifiers. These classifiers have new labels attached to them which are then exchanged in a community of agents leading to a common language. This language then allows agents to use these labels to denote common objects or properties of objects. This is what symbol grounding from an engineering A.I. perspective is about, but we argue that something is missing in this type of brute categorization. Each label, that is used in an agent, is associated with a kind of classifier based on external stimuli and is thus said to be grounded. The classifier generates some kind of perceptual category by using a prototype which gets refined along the way by playing language games with other agents. This causes them to align the labels and the generated prototypes for the categories. There is a clear division of labour here with respect to what the classifier does and the rest of the system that aligns the languages and captures the functional relations between the symbols in some kind of grammar (however rudimentary that may be). It is this division of labour between systems that is quite prominent in the work of Harnad. [12]

In a pure symbolic model the crucial connection between the symbols and their referents is missing; an autonomous symbol system, though amenable to a systematic semantic interpretation, is

¹In the intended philosophical sense [16]

ungrounded. In a pure connectionist model, names are connected to objects through invariant patterns in their sensory projections, learned through exposure and feedback, but the crucial compositional property is missing; a network of names, though grounded, is not yet amenable to a full systematic semantic interpretation. In the hybrid system proposed here, there is no longer any autonomous symbolic level at all; instead, there is an intrinsically dedicated symbol system, its elementary symbols (names) connected to nonsymbolic representations that can pick out the objects to which they refer, via connectionist networks that extract the invariant features of their analog sensory projections. [12]

There are two things going on in Harnad’s grounded agents that are quite distinct:

1. They categorize their environment in some way, thereby making it discrete, which allows for attaching some kind of labels to them. (I.e. identifying some color and inventing a new word for it.)
2. The functional role of the label (concept) plays some part in the mental circuitry of the agent. This allows for the “crucial compositional property”.

The first point is about brute categorization leading to some kind of perceptual category to which some kind of identifier can be assigned. This type of processing is usually performed by connectionist systems. Symbolic systems are commonly employed to model the second point, but how is this relation automatically formed?

There are fascinating parallels between this division of labour and *Conceptual Role Semantics (CRS)*. The so called two-factor account of CRS allows for the use of different semantic theories. One for the narrow scope, i.e. the inferential role that a concept plays in an agent and another for the wide scope, i.e. that to which the concept refers to (outside of the agent). Such a two-factor account allows for differences in individual concepts while they refer to the same entities. Conceptual Role Semantics is more extensively covered in section 3.5. Fodor and Lepore however objected to this so called two-factor-account of CRS, because they argued that there is nothing that “keeps the two together” (the narrow and wide scope theories) [10].

The division of labour, as argued above, makes this relation between narrow and wide scope essential for successfully using concepts in an environment. The large number of concepts presupposed by extreme nativism is essential for acquiring other concepts later on and additionally allows the relation between narrow and wide scope to evolve automatically. If we assume that connectionist systems can solve item (1) and symbolic systems can solve item (2), the engineering problem suddenly becomes a way to gain empirical insight into the philosophical problem.

1.4 Research questions

The thesis proposes a framework inspired by the theory Harnad puts forward in [11]. A two-way interaction between the two will be attempted, but not in the same way as suggested² in the article. In Harnad’s proposal the two systems are combined in a truly unified system where the structure of a connectionist system actually influences the operational semantics of the symbolic system in some unspecified manner.

We diverge from the Harnad’s approach in two important ways:

1. The structure of the neural nets will not affect, in any way, the operational semantics of the symbolic system, because the symbolic system is assumed to be immutable. This doesn’t allow the connectionist and symbolic systems to be merged into a single system, but does allow us to use existing unmodified symbolic systems.
2. The framework that is to be embedded in the agent does not perform actions. This is a job for the symbolic system, or something it delegates it to. This leaves the agent only with passive alignment of perception to mental structure (knowledge base, symbolic system, etc).

Even though these changes exclude the possibility of sensorimotor grounding, the strong decoupling is very much wanted from a software engineering standpoint and the alignment still offers advantages. At this point it is unknown if the approach can be easily extended to truly semiotic dynamics [36] including the acquisition of new concepts.

The proposed framework allows for *symbol grounding*, but does not offer a solution for the *symbol grounding problem*.³ Despite the great similarity in name they are quite different. Symbol grounding entails that an agent can relate its knowledge to reality in some way, but the symbol grounding problem is much stronger than that. The symbol grounding problem needs sensorimotor interaction with the environment and something different from mapping symbols to other symbols (connectionism is such an alternative) in order to ground them. Grounding symbols, as proposed in this thesis, is a big step forward in terms of the embodiment of knowledge based systems, but is still susceptible to Searle’s argument.

The major research questions, with respect to the challenges sketched, are:

1. Is it possible to align arbitrary sensor data with some symbolic description? And if so, what are the limitations and preconditions?
2. Is the functional/inferential role of a concept sufficient to acquire the “ability to categorize”? What needs to be assumed about such a process to be a viable explanation of an agent that grounds concepts in its knowledge base in accordance with a hybrid architecture?

²The article more actually hints at a possibility

³I would like to thank the audience of the Seminar A.I. meeting for contributing to this important distinction.

This thesis will first sketch the problem that is to be solved by defining a toy problem in chapter 2 to make it manageable. Then some philosophical introduction will be given in chapter 3 to place the framework in its proper philosophical context. This will make the strengths and weaknesses clear and offer philosophical insight into the problem at hand. Then in chapter 4 a very short introduction to some machine learning techniques will be given. This will make sure that the reader has sufficient information to fully understand the framework proposed in chapter 5. Results will be shown and discussed in chapter 6 in order to illustrate the effectiveness of the approach, then followed by the conclusion and discussion in chapter 7.

Chapter 2

Toy problem

2.1 Parallels and components

A toy problem or its philosophical counterpart (the thought experiment) is a way of quickly testing an idea. All of the toy problem's dimensions such as efficiency, consistency of the reasoning applied and an indication of the feasibility can be acquired without all of the intricacies of the full blown problem. A given solution's advantages and drawbacks can be easily investigated in such a context. Toy problems are usually more easily understood and problems are discovered at an earlier stage when compared to the full blown problem.

The following toy problem may look deceptively simple, but has some features that make it pretty hard to solve. The toy problem's job is to expose the essential features of the class of problems that the proposed framework tries to solve. The toy problem mentioned here is inspired by a classical challenge for machine learning systems employed in the domain commonly referred to as *Machine Vision*.

The task is about trying to solve a problem using the following components:

- A symbolic system.
- An untrained subsymbolic/connectionist unsupervised learning system that discretizes the input.

The symbolic system encodes domain specific knowledge in some kind of structured representation. One cannot say much about such a symbolic system other than that the arbitrarily chosen symbols encode information by having certain relations with other symbols. How these symbols (not their relations) relate to the outside world is unknown, unspecified and generally not a part of the symbolic system (for good reasons). The relation between symbols and what they refer to is of course not directly available, but only through some kind of sensor or sense organ.

Connectionist systems are quite successful at fulfilling the role of preprocessing the input from such a sense organ. They, for example, reduce the input to

something more manageable [18] or assign category labels to presented inputs. This reduces and stabilizes the information into something a symbolic system can process.

In this setting the toy problem is about bridging the gap between the analog (continuous, noisy, varied) input that the sensors receive and the discrete symbolic representations in the symbolic system. This problem can be solved by combining the two classes of systems into a single whole.

The idea of connecting the two different classes of systems has many concrete examples such as:

1. Automatically form a connection between words captured in a grammar and a speech signal.
2. Process scene graphs [43] and their relation to the outputs of a connectionist system that can recognize and locate objects in a raw image.
3. Translate unknown symbols from a language, when only given a grammar and a semantic knowledge base describing what the unknown language presumably encodes.

2.2 Recognizing arithmetic

The concrete toy problem on which the proposed framework is to be tested, is about correctly classifying handwritten digits as being members of certain categories (the numbers they represent). There is extensive literature on this subject [24] which we will not discuss in great detail, but the problem is regularly used as the proving ground for machine learning techniques. Inspired by this challenge the proposed toy problem builds on the knowledge gained from these studies and turns up the notch a bit concerning the difficulty of the task.

The extended toy problem is roughly the following:

There is a set of randomly generated handwritten, simple, correct, arithmetic expressions such as:

- $2 + 2 = 4$
- $7 - 9 = -2$

These expressions are only accessible as a list of bitmap images and not as text, in the sense that the label (the digit or any other symbol the bitmap images represent) is unavailable. The bitmap image serves as an analogue to low-level sense data received from the environment by, for example, a camera, but in general any type of sensor. The bitmap image introduces noise and variation in the input in a natural way, so that rote learning¹ is out of the question.

¹Every time you encounter a problem and solve it (or get the solution) store the problem-solution-pair in a table. Future instances of the problem are 'solved' by performing a lookup of the problem in the table of problem-solution-pairs.

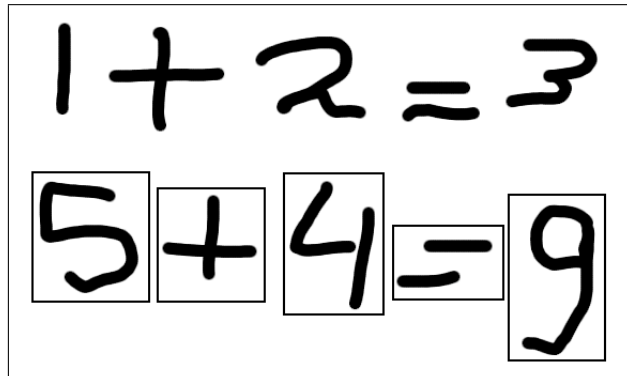


Figure 2.1: Two examples of handwritten arithmetic expressions. Above: raw image without segmentation; below: with segmentation. The segmentation is visualized as bounding boxes around each digit or operator.

In order to efficiently compose a large random set of arithmetic expressions the bitmaps of complete expressions are not directly offered to the system, but a list of its components is. For every handwritten symbol there exists a number of bitmap images representing that symbol [23] (they are labeled as such). By augmenting this commonly used data set with operators such as ‘+’, ‘-’ and ‘=’ the (analog) alphabet is complete for generating simple arithmetic expressions. We can do this, because the segmentation and order of the full expression in its constituent parts (handwritten symbols) is assumed to be given.

An expression is then generated, for example by selecting two random numbers A and B , an operator O and the result as computed by a calculator as C in an expression such as:

$$AOB = C$$

Each value (A, B, C, O) is then replaced with a random bitmap image that represents the value. (Multiple images are required for numbers larger than 9, because numbers larger than 9 consist of a sequence of digits). The total number of syntactically valid expressions that can be generated of this form is roughly: $10 * 2 * 10 * 1 * 10 = 2000$ of which only a subset is arithmetically valid².

Additionally, there is a symbolic system available that can give evaluative feedback about a possible interpretation of a complete sequence of bitmap images, but only in the form of a single scalar. For example the system could return ‘0’ for an incorrect sequence of symbols and ‘1’ for a correct one. The expression “4+8=12” would be assigned a value of ‘1’ by the symbolic system, whereas the expression “4+2=42” would get a value of ‘0’, but things are not as straightforward as these examples seem to suggest. An expression such as “3+5=8” will also result in positive feedback, even if the system incorrectly recognizes a handwritten ‘3’ as five and a ‘5’ as a three. While perfectly valid in this single instance, the mapping is of course not valid for all expressions and is therefore a local optimum. The same reasoning applies for the expression “4+2=42” that

²Intentionally ignoring reification (introducing semantic notions in syntax)

can result in a positive feedback of '1' when the (handwritten) '4' is interpreted as a zero resulting in the correct expression "0+2=02". The same goes for additionally mapping the '+' to a zero resulting in "002=02", another (locally) correct mapping.

The proposed system does not deal with locating stimuli in a complex input, but only with its identification. It's assumed that the perceptual system³ that is optimized for the task provides it with computational percepts. These may be annotated with spatial or temporal coordinates, but are not interpreted or even used by the part of the agent that maps these percepts to symbols. The annotations may be left out and will only reappear in the expression fed to the symbolic system. The system just deals with interpreting the percepts and linking them to symbols on the basis of the functional relations between them.

To summarize: On the one hand there is raw data representing the handwritten symbols which are processed by a system resulting in a kind of 'computational percepts'. On the other hand there are unambiguous digits as represented in the symbolic system. What is the relation between the categories defined by the connectionist system in relation to the digits and other arithmetic symbols, with a well defined semantics in the symbolic system? I will now turn to some philosophical background of this question in order to gain insight into the possible solutions.

³Though this could also be multiple systems each optimized for a specific goal

Chapter 3

Conceptual nativism & hybrid systems

The toy problem is quite trivial to begin with, but can have serious implications for certain philosophical positions. It is specifically relevant for those areas that revolve around questions such as: “How do symbolic reasoning and sensor data interact?” and “What are the limits of concept acquisition considering various assumptions?”. This chapter aims to achieve two things:

1. The problem and its proposed solution will be positioned in a philosophical tradition giving some insight in its strengths and weaknesses as have been discovered in this rich tradition.
2. Successfully solving the problem exemplified in the toy problem strengthens the arguments for the use of conceptual nativism with respect to artificial knowledge based systems. It will also try to counter some arguments raised against it and against two-factor Conceptual Role Semantics.

Keep in mind though that the main aim is to link the proposed architecture to its proper philosophical context and not to propose a full theory of concept acquisition for all types of intelligent systems¹.

3.1 Symbolic limitations

Syntax and semantics are the two prominent dimensions in language research and all its aspects. Syntax governs the structure of language and specifies the correct concatenations of components. On the other side semantics deals with the interpretations of a subset of correct sequences of symbols. For formal languages there necessarily exists a semantic interpretation for every well formed sequence of symbols. In practice syntax and semantics always go together. Every expression in natural language has a meaning (either intension and or

¹Humans included

extension), even such diverse things as poetry and stories set in a world of fiction. The two can be artificially separated such as to get a syntactic language with no meaning. Take for example the infamous ‘abaab’-type languages used in the teachings of formal grammar. These do have structural properties, but no semantics. The type of semantics of computer programs (operational semantics) is said to be ungrounded. Here ungrounded means that they are restricted to making statements about internal structures and, as a result, they cannot reach out and make statements about external world in any true way. In other words: it is impossible to make genuine statements about things outside of the system.

The groundedness of symbols only exists in the interpreter and not within the system itself.² It is a sign³ and nothing more, it has no intrinsic meaning. Symbolic systems are said to be ungrounded, because they only concern the manipulation of *meaningless* tokens. The Chinese room example is a famous philosophical example meant to illustrate this.

Imagine a native English speaker, let’s say a man, who knows no Chinese locked in a room full of boxes of Chinese symbols (a data base) together with a book of instructions for manipulating the symbols (the program). Imagine that people outside the room send in other Chinese symbols which, unknown to the person in the room, are questions in Chinese (the input). And imagine that by following the instructions in the program the man in the room is able to pass out Chinese symbols that are correct answers to the questions (the output). The program enables the person in the room to pass the Turing test for understanding Chinese, but he does not understand a word of Chinese. [17] (page 115)

The important thing is that the man in the room has no idea what the Chinese that he receives and produces means. He just looks up symbols in a database and writes other symbols down that are mentioned in the database. He knows nothing about the meaning of the symbols yet produces correct results. In Searle’s view this is what symbolic computation is all about: just rewriting signs into other signs (though perhaps symbols from the human perspective) without any notion of what is actually written about. The meaningless symbols are said to be *ungrounded*, i.e., they have no relation to reality.

This is the problem concerning symbol systems as exemplified in Searle’s Chinese room thought experiment [34]. Harnad [12] made further contributions clarifying the issues surrounding Searle’s thought experiment and, more importantly, offered a way out by grounding symbols in perception, thereby stepping ‘out of the loop’ of rewriting meaningless signs into other meaningless signs⁴. This escape-plan heavily depends on the effects of the intrinsic properties of connectionist or subsymbolic systems on the interpretation of symbolic expressions.

²Though such a system does have an operational semantics for it

³In the sense of Peirce [2] a sign is a type of signifier, such as a written word, a spoken sentence, smoke as a sign for fire, a pictogram, etcetera.

⁴The distinction between signs and symbols is intentional

3.2 Categorization

The world doesn't come to us in distinct objects, but the perception of distinct objects is a rather active process on the side of the interpreter. In essence all you get is one big sensory experience. Hopefully these experiences contain some detectable patterns or regularities which can then be used to carve up reality into its constituent parts. This perhaps with the help of some innate process(es) allowing more complex forms of computation because of the information reduction caused by categorization. Categorization seems to be an essential precondition for exhibiting cognitive behavior. [13]

Harnad then argues that “our ability to discriminate inputs depends on our forming *iconic representations* of them” [13] (*italics added*). To categorize a sensory experience and to map it onto one of the icons one only needs to do a comparison of the sensory input to the iconic representation. The likeliness of the two then determines which iconic representation accompanies which input. The problem is then; how are these iconic representations formed? And what properties should they have? It is unlikely that they are in some way a concept in the classical sense, because that introduces a host of other issues. For example: What are the invariant features that reliably select the dog concept? A dog with 3 legs, 1 eye, no hair, etc is still a dog. This all points to the idea that mental concepts should, in some manner, be very rigid and at the same time flexible.

Harnad makes a case for a process that uses iconic representations that pick out the *invariant features* (those properties which maximally differentiate one category of inputs compared to others) of the sensory projection that “reliably distinguish a member of a category from any nonmembers with which it could be confused” [12]. The symbol-grounding-problem makes it necessary that these iconic representations are non-symbolic. Symbolic iconic/categorical representations would reintroduce the symbol-grounding-problem on a different layer. An example would be a neural network that correctly classifies some inputs. It stores the invariant features implicitly in its weights.

[The connection to the objects that iconic and categorical representations pick out is] based on the relation between distal objects, proximal sensory projections and the acquired internal changes that result from a history of behavioral interactions with them. Nor is there any problem of semantic interpretation, or whether the semantic interpretation is justified. Iconic representations no more “mean” the objects of which they are the projections than the image in a camera does. Both icons and camera-images can of course be interpreted as meaning or standing for something, but the interpretation would clearly be derivative rather than intrinsic. [12]

In other words: the step of interpreting the signal in some way is prevented and the symbol grounding issues raised by Searle are non-applicable by these types of representations. The resulting categorical representations (hopefully) rigidly categorize/pick out classes of objects, but do not *represent* them intrinsically. Once a basic repertoire of various identifiers has been found one could associate

these with symbols in a language. New symbols (concepts) are to be composed of elements from the ‘basic repertoire’ and inherit the grounding from this basic set. Concepts are said to be an important component of cognitive systems that need to make the leap from categorization to identification.

Let’s first illustrate the difference between categorization and identification with a small example: Let’s say that in some culture red marbles are gifts of affection and blue ones are meant to insult the receiver. Now I can perfectly categorize these two types of marbles by putting all the blue ones in one basket and the red ones in another one. This categorization however is not dependent on the full functional role that these special marbles play in my mind. In my mind these two types of marbles have different functional roles, because one means insult and the other affection, something not captured in brute categorization.

This crude “functional role”-less type of categorization is what connectionist systems usually do, but in humans the categorization seems almost intrinsically connected to its functional role.⁵ Concepts are said to play an important role in this process of categorization.

3.3 From symbols to concepts

So it seems like complex cognitive systems (such as humans) do something more than just rewriting symbols. They’re said to be able to use grounded concepts to reason about the world, but what does that actually mean? What should the word ‘concept’ mean? And why is it grounded?

There are many competing theories of what concepts are. They are roughly summarized as follows [26]:

- Classical theory
A concept must decompose into a set of concepts that express necessary and sufficient conditions for the application of the concept. *Birds are animals with beaks, feathers, they can lay eggs and have the ability to fly.*
- Prototype theory
A concept must decompose into a set of concepts that express statistical conditions that govern the application of the concept. *Birds statistically are animals with beaks, feathers, the ability to lay eggs and the ability to fly, but none of these properties are essential.*
- Theory-theory
A concept participates in an inferential system of some sort. The concept is inherently connected to the other concepts that make up the system. *Birds are things that play some inferential role with other things and properties such as animals, beaks, eggs and the ability to fly.*

Now the theory-theory-type definition seems a bit circular at first sight, but it is actually a functionalist definition. Functionalism is the view that what makes

⁵This intrinsic connection between categorization and identification is similar to what Har- nad hints at when speaking of symbolic systems that are influenced some intrinsic property of connectionist sensors [11].

something a mental state is not its structure, but the role/the function that it plays with other mental states. The identity of a mental state is completely determined by its causal relations to sensory experience, behavior and other mental states. Defining mental states in such a manner allows for multiple realizability, because the ability to think and experience is then not a property of the specific biological neural structure of our brain, but a result of how it functions in the sense of its interacting parts. In the same way an artificial brain can be created, that can sustain the same level of cognitive performance as a human, if it has the same functional architecture. At least this follows from the (machine) functionalist view.

Another important ‘concept’ in the philosophy of mind is the ‘Representational Theory of Mind’. This doesn’t automatically follow from functionalism, but is quite often mentioned in one breath. It’s the idea that the mind forms representations of the external world and other mental contents. Cognitive processes then become operations performed on these representations. Note how this compares to symbolic system as previously mentioned. The representational theory of mind does not state by what process these states come to be and there are many competing theories concerning that, but it does raise issues about what these representations look like. Are they some kind of concepts? And if so which concept model is ‘right’?. Although much more is unspecified (and needs to be for such a general theory) there are some basic characteristics of the mind that are representational in nature (or at least, are conceptualized as such) that compare well with the properties of natural language.

Each of the concept-theories has theoretical difficulties and it has therefore led some philosophers to abandon all of them and opt for something completely different. In the sense of Millikan [30] concepts are abilities, more specifically: “the ability to reidentify”. Additionally it is presupposed that lexical concepts have no internal structure, making concepts atomic. This theory of concepts captures the functional relations with other concepts that these concepts play in mental states as well, but we want to treat the role that a concept can play as a mental state separately (for reasons given below). That’s why the ability to consistently categorize a certain class of objects will be called “the ability to categorize” instead of “the ability to reidentify”. The correct categorization of a concept is thus defined as an ability that does need to be available (but not represented in the classical sense) in some form of representational medium to allow for a complex functional role.

3.4 Extreme nativism

Fodor is a proponent of conceptual atomism and has a functional view on the content of concepts. There is a tension in the sense of the nature-nurture debate concerning concept acquisition. When adopting the functional theory-theory concept position the content of a concept becomes dependent on its inferential role, but that seems to entail that some concepts need to be hardwired into the intelligent system. This nature-nurture-tension has led Fodor to adopt the position of extreme nativism with regard to concepts. The main arguments for taking this stance are as follows:

1. A large number of concepts isn't definable [27] by means of reference to other concepts.

... suppose that the concept FATHER is the concept of a male parent and that the concept has the structure MALE PARENT, that is, it is literally composed of the concepts MALE and PARENT (and whatever logico-syntactic concepts may be involved). In this case, one can imagine that the acquisition of FATHER proceeds by noticing that some parents are male and by constructing a complex concept to reflect this contingency, namely, MALE PARENT (= FATHER). Notice that, in this way, the learning of FATHER takes place only on the condition that the agent previously possesses the concepts MALE and PARENT. Turning to the component concepts, MALE and PARENT, we can now ask the same question about how they are acquired. Perhaps they too decompose into simpler concepts and are acquired in much the same way as we are supposing FATHER is acquired. Yet clearly this process has to stop. [21] (page 63)

In a sense this is a bootstrapping problem. The complexity of this bootstrapping problem for 'common knowledge' (whatever that means) is perhaps shockingly visible in the Cyc project⁶ [28].

2. Undefinable concepts aren't learnable (this requires further argumentation as will be provided below)

These are pretty big statements to make, but Fodor has reasons for doing so. It depends, in large part, on how the mechanism of concept acquisition is defined. Fodor defines it as a type of inductive learning. This process of inductive learning is well illustrated by giving an example of a so called "concept learning experiment" where subjects learn to acquire new concepts:

... the experimenter has a particular concept in mind, which is labeled with a novel predicate, say, 'urg'. Subjects are asked to sort various stimuli according to whether they are 'urg' or not, where all they have to go on is the feedback that the experimenter provides after each trial. For example, if 'urg' is the concept green, then when the subject says that a card with a green circle on it is 'urg', she'll be told that she is right. And if she says that a card with a red circle on it is 'urg', she'll be told that she is wrong. And so on. Eventually, if all goes right, she'll come to reliably sort cards according to whether they are 'urg'. [22] (page 29)

This formulation makes it a representational theory. New concepts that are to be learned are defined by properly describing the 'to be learned concept' using existing ones. This all sounds a bit circular and that's exactly the problem

⁶A long term project (started in 1984) that tries to formalize common knowledge using symbolic reasoning tools.

Fodor tries to address. It all boils down to: What do you need to bootstrap and what can be inferred from the data itself? This is the part in extreme nativism that becomes sticky. Extreme nativism states that we are bootstrapped with most of our lexical concepts, in stead of just a few.

As argued above a large number of hypotheses cannot be formulated by means of existing concepts and therefore cannot be learned as defined above. Since these concepts cannot be learned, but are used successfully nonetheless, these concepts (and many others) have to be innate (or a bit more considerate: the mechanisms for acquiring these concepts are innate.). Furthermore (assuming a “language of thought”) the expressive power of the cognitive system isn’t enlarged by the acquisition of a new concepts, but only its efficiency (the chunking metaphor from SOAR may be insightful [20]⁷). This imposes certain hard limits on the expressivity of human thought.

We can lose some of the stickiness by presupposing that concepts can be acquired and are structured, but are not necessarily defined. (It is quite counterintuitive that we have some kind of definitions of many previously unencountered concepts stored in infants.) In this manner, the problems concerning specification in a sufficiently rich mental representational system are avoided. For example: One can use a neural network to recognize a certain (analog) signal. Here the criterion for recognition is defined as responding to the signal in a correct manner which has been specified in advance. The neural network is structured, or becomes structured during learning, in such a way that given a certain class of inputs, the intended outputs are triggered without explicitly specifying the concept it has learned the right responses for. The neural network is structured but there is no explicit definition of what has been learned. So in the human case: The brain has certain structures that are needed to use certain concepts without that entailing that we have an explicit definition for them. [11]

It’s important to note that these innate concepts aren’t readily accessible by the subject, but need to be triggered by the right experience. Otherwise, people would be able to use concepts without ever being exposed to them or learning them otherwise. Each individual is said to ‘enable’ parts of the total conceptual space available. This triggering doesn’t need to be the straightforward grabbing of ideas from the outside of Plato’s cave or Frege’s “Third Realm” (directly accessible and known, instead of mentally construed), but should rather be seen as the “unfolding of some internally governed mental structure” [26] which is or can be dependent on a shared neurobiological basis.

Though extreme nativism seems quite troublesome when trying to explain highly arbitrary cultural concepts (like pingpong-ball or flippo) it makes good sense to capture many AI systems in use today, specifically those that fall in the realm of knowledge based vision systems [8] and the like. See [6] for a more philosophical treatment of the subject. In these cases the knowledge is truly innate and concepts do get picked out in reality. Concept acquisition, in such systems, is not problematic, because these are generally static systems with respect to their knowledge base.

⁷Whenever the decision cycle in a SOAR-program returns the result for a super goal, a new rule is created whose contents are the elements of the returned result. This new rule is called a “chunk”. The result is a reduction in the number of rules required to achieve a super goal.

3.5 Differentiating concept competence

Let us assume that extreme nativism is actually the case. Is there a way to weaken the supposed implications of this position with respect to the bootstrapped concepts? I will argue that this is possible by separating two different aspects of concepts that are frequently lumped together. The elements of the resulting bipartition will map nicely to connectionist and symbolic systems.

We can differentiate the idea of ‘concept competence’ (successfully using a concept) in two important ways:

1. “Ability to categorize”

The step from undefinable concepts has been made by Fodor, but doesn’t necessarily have to be the case for brute categorization. When taking the position that “having a concept” means “having the ability to categorize” things change somewhat. This doesn’t presuppose a representation of a concept (classical sense) in order to use a concept effectively, but it only requires the ability to do so. A concept need not be explicitly specified (in the sense of some connectionist system), but can ‘pick out’ a class of referents rigidly nonetheless. The ability to categorize may be perfectly innate as caused by a genetically shared perceptual system which pre-structures input, without that implying that the inferential role of the concept is innate as well.

2. “The functional role of a concept”

Linguistic concepts transferred through reference borrowing are a good example of this as illustrated in [30]:

... and, say, of African dormice. There, I just handed you a concept of African dormice, in case you had none before. Now you can think of them at night if you want to, wondering what they are like - on the assumption, of course, that you gathered from their name what sorts of questions you might reasonably ask about them (animal questions, not vegetable or mineral or social artifact questions). [30] (page 64-65)

Such concepts do have a functional role, but do not necessarily specify the extension of the concept.⁸ Purely functional role concepts can have a kind of slot available that enables their ability to (re)identify the concept in the external world.

Conceptual Role Semantics (CRS) [3, 41] is a metaphysical theory about semantics. It states that meaning is functional in nature. The specific content of a mental state is dependent on the *conceptual role* that it plays in certain processes. This is in contrast with the *functional role* which may include non semantic causes and effects such as happy thoughts that promote good health and or improve concentration. For example, the meaning of a certain cultural artifact can only be understood in the context of that culture. So in order to

⁸I’m intentionally rejecting the view that language should also be seen as an environment and the mere exposure to a word constitutes an ability to reidentify

truly grasp the meaning of such an artifact you need to acquire its functional relations.

CRS offers a framework in which to express these functional relations for concepts (and other mental states). It defines the content of concepts as the inferential role that they play in the representational system and the relation that they (may) have to the external world. This two-factor account is needed to counter arguments raised by Putnam [31]. He argues that the meaning of many natural kind concepts, such as water and diamonds, depends, in part, on something more than just the internal representation. Differences in the external environment can cause the meaning of two identical functional states to be dissimilar. The two-factor account of CRS makes a distinction between an internal (narrow) type of meaning and an external (wide) referential/truth-theoretic kind of meaning. This will allow us to assert that two mental states that play the same functional role are identical in the narrow sense while differing in the external sense, because of differences in the environment.

A functional theory about concepts such as CRS almost automatically introduces problems with respect to bootstrapping because of the mutual dependence of the meaning of concepts to one another. A solution to the problem of concept acquisition has been given in [22] where Block solves the puzzle of concept acquisition with respect to nativism in the following manner:

Concepts that lack compositional semantic structure may be learned so long as they are acquired by a process that establishes not just their individual inferential roles but also the inferential roles of all of the concepts to which they are constitutively related. So long as all of these inferential roles can be brought about together, all of the implicated concepts can be learned together. [22] (page 47)

The narrow type of meaning corresponds nicely to the previously mentioned functional role of a concept and the wide aspect of meaning to the *ability to categorize*. Reviewing the nativist position in the light of the two-factor CRS shows some interesting properties when applied to AI systems. Connectionist systems are generally employed to realize this ability to categorize, whereas symbolic systems are better suited at capturing complex functional relations.

Pure symbolic systems are said to have only functional roles (operational semantics), thus giving rise to the symbol grounding problem when trying to use them to relate symbols to the world. Connectionist systems are great at avoiding this line of reasoning and can rigidly categorize objects, but aren't as well suited to representing complex functional roles as a symbolic system is.

In a situated system [38] the two elements of a concept need to go together, because concepts represented internally need to be related, grounded, in the environment in which the system functions. When the system is not situated, there is no environment and therefore the "ability to re-categorize" can be left out without penalty. In this case, signs are directly activated, needing no interpretation of the input.

Fodor and LePore objected to the two-factor account of CRS, specifically on what keeps the two parts together [3]. This thesis will defend the two-factor-

CRS by showing how the narrow/internal functional relations restrict the interpretation of the input. Though one could state that some strange correlation between the internal and external semantics exists, this thesis will only address the interaction between the two from the perspective of the interpreter in a behavioristic manner.

The connection between the connectionist and iconic/symbolic representations is a closely related problem and many parallels can be drawn between the philosophical problem at hand and the interaction of connectionist and symbolic system. On the one hand one may wish to combine the two types of systems to escape from the “The Chinese/Chinese Dictionary-Go-Round” [12] (Where one is just manipulating ungrounded symbols without ever grounding them in perception) without giving up complex function roles. On the other hand it raises problems such as: how are categorical/iconic-representations formed? And how should the connection between these abilities and symbolics system that capture complex function relations be made?

The framework that will be described in chapter 5 will be in line with Fodor’s nativist stance on concept acquisition. All concepts are innate in the symbolic system and recognized within the environment. The relation between the internal and external semantics is not something given in advance, but something that evolves. The subsymbolic/connectionist system then, in a way, *triggers* the correct concept (symbols are assumed to play the role of (atomic) concepts in symbolic systems) in the symbolic system and enables its functional role. The connectionist system could do this by forming iconic representations and matching the input with them.

Chapter 4

Machine learning

4.1 Introduction

Machine learning covers a large area of AI. It contains a large amount of approaches that range from rule-based reasoning systems and fuzzy-logic classifiers to connectionst neural networks, to just name a few. Machine learning plays a prominent role in this thesis, because it is used to specify a possible solution to both a common design practice and a philosophical question.

This chapter will introduce the various components used in the proposed framework introduced in chapter 5 and will shortly summarize the important properties of each. It will begin with the difference between supervised and unsupervised learning methods and will build on that introducing Genetic Algorithms as a semi-supervised learning method and Adaptive Resonance Theory as an unsupervised one.

4.2 Types of learning

4.2.1 Supervised learning

Supervised learning methods are used in tasks where the correct answer for a learning example is known. A well known example is recognizing tanks in images. Here you feed a supervised machine learning system a large number of tank images and non-tank images, each labelled as such. After repeated exposure the system can correctly label unseen images which include a tank or not.

Supervised learning methods need to be presented with a valid input-output-pair. The output to which it should be mapped can be a continuous value (regression) or a class label (classification). The supervised learning method tries to determine the mapping from any valid input to any valid output given a number of so called ‘training examples’. Performance is usually evaluated by using a previously unseen (by the algorithm) test set and the difference in

results is measured by means of comparing the results of the labeled data in the test set with the output of the supervised learning method used. Or to frame it differently: it has to generalize given a ‘few’¹ training examples. The psychological equivalent of this type of learning is called ‘Concept learning’ as described in section 3.4.

Supervised learning roughly follows these steps:

1. Create an example set with inputs and the corresponding correct class labels.
2. Divide the example set into a training and test set
3. Train supervised system using the training set
4. Evaluate the performance of the trained system on the test set

A common problem with supervised learning systems is overfitting. Overfitting happens when the learning mechanism treats the training set as the complete problem set. Thereby excluding unseen instances of correct input-output-pairs as based on the underlying generality of the problem set. Usually this is not really prevented (as in changing the algorithm), but it is worked around by periodically testing the trained system on a test-set. When performance on the test set continues to drop training is stopped, because of supposedly overfitting taking place.

4.2.2 Unsupervised Learning

Unsupervised learning methods don’t need any class labels for the input presented to them, but only inputs. Unsupervised learning methods cluster the data presented in the dataset into categories. This clustering is only performed by means of discovering the intrinsic structure within the data itself. Unsupervised learning methods get no correct labeling or any other type of feedback on their performance. In the end, after learning stabilizes or stops, the experimenter can attach meaning to the categories formed. Meaning could here be interpreted as: What does the cluster represent. E.g.: What are the attribute values that constitute the center of a cluster? This is in sharp contrast to supervised learning where the meaning of the output is known beforehand and explicitly given.

Examples of unsupervised learning techniques include “Adaptive Resonance Theory” [5], “K-means clustering” and the “Self organizing map” [18]. The advantage of these techniques is that they’re completely model-free and can discover patterns in data that would have been missed if the end result had been explicitly specified as with supervised learning. Unsupervised learning methods can be used to bridge the gap from complex inputs to abstract outputs, because they automatically group the inputs into abstract clusters, thereby reducing the complexity of the data. This more abstract and reduced data set is then easier

¹Which could easily include thousands of examples and is completely dependent on the problem size

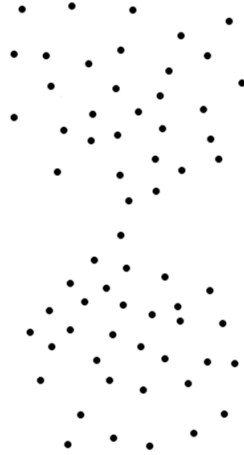


Figure 4.1: An example of “touching clusters” here illustrated by two clouds of points touching each other in the middle.

to match on the abstract outputs than with a supervised learning technique. In a sense it is reducing the dimensionality of the data (from the perspective of the input fed into the supervised learning system) in the same way as Kohonen Maps do [18].

Figure 4.1 illustrates an unsupervised task that people can do virtually without thinking² and usually can’t explain how they solved it, because the solution just seem to ‘appear instantly’ to them. Unsupervised learning, assuming the right parameters are set, will cluster these two point clouds without having knowledge of the number of clusters given. It is a so called unsupervised data-driven process.

4.2.3 Reinforcement learning

Reinforcement learning [37] is a learning method that lies somewhere between supervised and unsupervised learning. An agent doesn’t get the correct label for an input, but some indicator of how good the label (or in RL terms ‘action’) selected is. This indicator may be directly provided or with some delay, but the greater the delay, the slower the learning. Genetic algorithms work in a comparable way, but performance indicator value is specifically called the *fitness value*.

An example application of reinforcement learning is chess. You could feed some reinforcement learning system chess moves assigned with a value for how good the move is (for example if it leads to checkmate, loss of a queen, etc) or just a single feedback at the end of a game, i.e. -1 for losing and +1 for winning the match. In the long term the algorithm learns which actions to take in order to

²Naturally conscious processing is meant here

maximize the reward gained. The system learns the best move not by example or deduction, but by maximizing the long term reward.

This machine learning method of only providing evaluative feedback is quite distinct from supervised and unsupervised learning.

4.3 Genetic algorithms

Genetic algorithms are inspired by the principle of evolution by means of natural selection. Genes encode for phenotypes which have an evolutionary advantage or disadvantage. Individuals with more of an evolutionary advantage during their lifetime in their environment are said to be fitter (have a higher fitness value), because they have better chances of getting offspring into the next generation. If this trend continues they will, in the end, dominate the population. Random mutation ensures that new possibilities for improvement are discovered and reinforced by means of the fitness and resulting selection pressure within the population.

'Individuals' will be called 'solutions' for the remainder of this thesis. A 'population' will be called a 'pool' (a list of solutions ³). This will hopefully prevent confusion, because the scope of this thesis reaches beyond the regular domain of GA and may include multiple agents. Mixing agents and individuals would become confusing, therefore a different vocabulary was chosen.

Solutions (individuals) with higher fitness values have greater chances at getting offspring. Recombination combines the two different, relatively successful, solutions and creates offspring that inherits parts from both successful parents. Given that the genes are not incompatible the fitness of the offspring rises even more, and so on.

Listing 4.1: A genetic algorithm in pseudo-code

```

1 pool = Pool.new(n)
2
3 generations.times do
4   pool.each {|solution| solution.fitness = evaluate(solution)}
5
6   newpool = Pool.new(0)
7   n.times do
8     solution1 = pool.get_selected_solution()
9     solution2 = pool.get_selected_solution()
10
11     newsolution = Solution.recombine(solution1, solution2)
12     newsolution.mutate()
13     newpool.add(newsolution)
14   end
15   pool = newpool
16 end

```

Some additional comments about this pseudo-code:

³it's not a set in the mathematical sense, because the same solution may appear more than once

- At line 1 a solution pool is initialized with n random solutions
- Line 4 calculates the fitness for each solution
- Line 8 and 9 select a solution from the pool. Solutions with a higher fitness value have a higher probability of being selected.
- Line 11 implements the recombination of the two selected solutions, resulting in a new solution

The exact implementation of the selection mechanism (lines 8 and 9) is quite important and comes in the following flavors:

- Boltzman selection (also known as Softmax action selection)

$$\frac{e^{f(i)/\tau}}{\sum_i e^{f(i)/\tau}}$$

τ here is called the temperature which controls up to what extent a fitness score of solution i determines the probability of getting selected. Increasing the temperature decreases the effect of fitness on selection.

- Rank Selection

Every solution gets grouped by fitness value and these groups are then sorted. The group with the highest fitness value is assigned 'rank 1', the second best is assigned 'rank 2' and so on. The selection probability is then inverse proportional to the rank. The exact relation between rank and selection probability is left unspecified, but in general this setup is used to give the best solutions only a moderate advantage.

- Steady-State Selection

In the regular generation GA the new pool consists entirely of new offspring (elitism is an exception to this) whereas in Steady-state only a small portion of the pool is replaced with new offspring. This strategy can be employed in systems that need to do incremental learning.

- Tournament selection

A group of solutions of size n is taken from the solution pool. The fitness values for these solutions are calculated and the one with the highest fitness is added to the new generation. By variation in the size of the pool selection pressure can be increased or decreased.

- Elitism

The best solution (not specifically limited to a single one) always makes it into the next generation without being affected by mutation or recombination. That way the best solution discovered during the run of the GA is not lost, but it is even guaranteed to be in the resulting pool when the algorithm has finished running.

The parts that make up the components of a solution are sometimes called ‘building blocks’. A recombination operator is said to recombine these building blocks. The key to understanding why GA work and how the different components interact (selection pressure and mutation and or recombination) is captured by the *Schema Theorem* as summarized in [39]:

$$m(h, t + 1) > m(h, t)\phi[1 - \epsilon(h, t)] \quad (4.1)$$

Where $\phi(h, t)$ being the reproduction rate of generation h at time t , $\epsilon(h, t)$ the disruption factor⁴, ϕ is the growth rate by selection only and $m(h, t)$ being the size of the solution. The equation captures that the number of separate building blocks increases when the number of new solutions created is larger than the number of solutions destroyed by disruption(recombination, mutation).

$$\phi[1 - \epsilon] > 1 \quad (4.2)$$

The equations seem to suggest that an increase in ϵ (disruption) can be countered by increase in ϕ (selection pressure), but things are not that easy. Large parts of the diversity in the solution pool will disappear in a short amount of time and the GA will get stuck in a local optimum if ϕ is increased too much in an attempt to lower ϵ . On the other hand, when ϵ is small the same risk of getting stuck in a local optimum remains, because larger leaps in search space do not take place and exchange of building blocks between solutions comes slowly to a halt.

We need to add some more sophistication to the Schema theorem to explain these effects which are related not to the increase in building blocks, but in the recombination of them. Having the buildings blocks somewhere in the solution pool is no guarantee for them getting recombined in the correct manner in order to achieve a satisfying solution. There needs to be a robust mechanism that stimulates (by means of selection) the recombination of partial solutions in order to a achieve a better one. Our recombination operator has to satisfy two conflicting goals:

1. Minimize ϵ
2. Maximize the exchange of building blocks and or mixing of them

Possible implementations for the recombination operator:

- 1-point crossover

The solution is divided in two parts for each parent which then get recombined resulting in two children.

⁴This sounds worse than it actually is, since mutation is the mechanism that generates new solutions with a possibly higher fitness value

- 2-point crossover

A start and end index is determined for the two parent solutions. The part that exists between these indices is swapped between the parents results in two new offspring

- uniform crossover

A child is created by walking along the two parent solutions while selecting parts from the one or the other with some probability p . Usually $p = 0.5$. In this thesis we will primarily look at *uniform crossover*, because there is little information available about how the parts of the solutions interact within the GA.

Linkage is the phenomenon whereby parts of the solution have some form of relation to each other that makes them a group. These groups as a whole have an increased probability of being passed to offspring. There are many ways to view these groups in the solution structure such as:

- Compact representation of the solution
- Parts of the solution which are close together spatially
- Parts of the solution which are similar

Thierens [39] quite thoroughly argued, using the bit-counting task for different GA, that linkage is essential for scaling GA to larger problems. When no linkage is assumed between parts of the solution either the population increases exponentially or the running time does, compared to the size of the solution. Either way, introducing linkage is practically a requirement for complex GA's that need good convergence results on lengthy solutions with possibly large data sets.

It is important to understand the inductive bias of EA techniques such that we have a good indication of its performance beforehand. If the chosen EA matches the structure of the problem then good results are more likely expected to be attained. This is because every algorithm makes certain assumptions about the distribution of solutions in the search space. For example they inductively create better solutions based on previous solutions (with of course some degree of probability due to mutation and selection for example).

An important subset of EA is the class of Multi Objective Evolutionary Algorithms (MOEA) [7]. This class is specifically tailored to solving optimization problems with multiple *conflicting* objectives. Normally GA's converge to a single solution, but it could be that there are multiple mutually incompatible solutions which require MOEA. MOEA can generalize to a non-conflicting single objective EA by reducing the size of the pareto optimal set to 1.

4.4 Adaptive Resonance Theory

Adaptive Resonance Theory (ART) is an architecture for unsupervised clustering algorithms first proposed by Grossberg [4]. Its first incarnation called ART-1

only deals with binary input patterns. ART-2 is an extension of ART1 that can deal with both binary and analog input patterns. ART-2 is also commonly referred to as *Fuzzy Art*.

On the one hand one wishes to retain previously learned information and on the other hand new information needs to be integrated into previously acquired information (the model). Problems occur when newly learned information interferes with the previously learned information. This is commonly referred to as ‘catastrophic forgetting’. It could interfere, for example, because the system only has a small amount of memory available to store its model, leading to a continuous trade-off taking place between storing new information and retaining old information. This tradeoff is called the *stability-plasticity-dilemma*. A system is called ‘stable’ if previously learned information does not degrade and ‘plastic’ if it can adapt to new information. The ART-architecture was designed with this dilemma in mind.

The ART-architecture needs no information in advance about the number of clusters, but it is dependent on a matching routine that can be easily adjusted. The adjustment is a trade-off between the number of clusters and accuracy. New clusters are added dynamically without invalidating previously learned information allowing incremental learning of both the number of clusters and their centers.

On presentation of an *input vector* the *templates* are sorted with respect to their activation value for the presented input. The templates are selected in ascending order and for each template the match between the template and the input is calculated. If the match-value is the same or higher than ρ (an indicator for how well an input matches a template) a template is said to *resonate* and is marked as such. The search then terminates and returns the template or an identifier for that template. After resonating, the matching template is updated towards the presented input vector in order to improve it for future input presentations. When all templates have been visited but none match, a new template is created and initialized with the presented input.

Listing 4.2: ART-2 algorithm

```

1 templates = Set.new(Template.new(), Template.new(), ...)
2 while (env.newInput?)
3   input = env.getInput()
4   add_new = true
5   templates.each { |template| template.activate(input) }
6   templates.size.times do
7     template = templates.sort_by_activation().first()
8     if (Match(input, template) >  $\rho$ )
9       template = template.update(input)
10      add_new = false;
11      break
12    else
13      template.activation = 0
14    end
15  end
16  templates.add(Template.new(input)) if (add_new)
17 end

```

Fuzzy set theory is used to calculate the similarity of an input pattern to a template and to calculate the so called *match* between the two.

The following operations are from fuzzy set theory and are used below to explain the exact implementation of the methods. Here is a small recap of the operators used:

$$\begin{aligned}\vec{a} \cap \vec{b} &= (\min(a_1, b_1), \min(a_2, b_2), \dots) \\ \vec{a} \cup \vec{b} &= (\max(a_1, b_1), \max(a_2, b_2), \dots) \\ \neg \vec{a} &= (1 - a_1, 1 - a_2, \dots) \\ \|\vec{a}\| &= \sum_{i=1}^D |a_i|\end{aligned}$$

There are several functions in listing 4.2 that will be defined in the following equations using the fuzzy set operators.

$$\text{activate}(\vec{input}, \vec{template}) = \frac{\|\vec{input} \cap \vec{template}\|}{\alpha + \|\vec{template}\|} \quad (4.3)$$

where $\alpha > 0$ is a choice parameter. α will trigger a template sooner for an input. Do note that this is different from actually *matching* a template.

The equation 4.4 captures the match-method.

$$\text{match}(\vec{input}, \vec{template}) = \frac{\|\vec{input} \cap \vec{template}\|}{\|\vec{input}\|} \quad (4.4)$$

The vigilance parameter ρ in equation 4.4 determines the error margin in matching an input to a template. When equation 4.5 is true an input is said to resonate with a template. Increasing the vigilance (ρ) will generally lead to an increase in the number of templates, because the matching is stricter.

$$\text{match}(\vec{input}, \vec{template}) > \rho \quad (4.5)$$

After an input resonates with a template, that template gets updated towards the input using equation 4.6. When $\rho = 1$ (equation 4.5) ART reduces to exemplar learning as each new input pattern will get its own template.

$$\text{update}(\vec{input}, \vec{template}) = (1 - \beta)\vec{template} + \beta(\vec{input} \cap \vec{template}) \quad (4.6)$$

β is the learning rate, $0 < \beta \leq 1$. Increasing the learning rate will result in faster learning (the template is updated increasingly towards the input).

Learning with $\beta = 1$ is often referred to as *fast learning*. This makes sure that even after just a single presentation of an input the template will match the input directly the next time. If β is significantly lower than 1 training a new template on a certain input takes more than a single update. Now it is common knowledge⁵ that for neural networks to converge, the learning rate needs to drop slowly during the last iterations. In ART this needs not be the case if complement coding is used as described by equation 4.7, it also prevents category proliferation [33](page 46).

$$(x = (\vec{x}, \neg\vec{x})) \tag{4.7}$$

Because templates are only updated when resonance with an input occurs interference between different templates is prevented. This is a problem that commonly plagues neural networks. Another advantage is that inputs which occur quite seldom are recognized accurately the next time, while in the neural network case the information frequently degrades with the number of intermediate updates. Incremental learning of sparse data amidst other high frequency inputs is possible.

⁵To be fair only in certain specialized academic circles

Chapter 5

The framework

The proposed framework will leverage the ‘building blocks’ (Fuzzy ART and Genetic Algorithms) introduced in the previous chapter. Being more of a framework than a problem specific implementation; the specific components used in this framework (such as certain sensors or a specific genetic algorithm) are interchangeable with more efficient versions. The concrete implementations are mentioned to illustrate and explore the various properties. It also makes it possible to empirically validate the possibility of solving the task defined by the “Toy problem” in an efficient manner.

A detailed description of the various parts of the algorithm is given, including various arguments for the large number of design choices made. Pseudo-code is used to give a concise summary of some of the algorithms.

5.1 Overview

The proposed framework tries to automatically ground symbols from a symbolic system in perception. Which symbolic system exactly isn’t really of interest, but the toy problem illustrates a concrete one to give a handle on things. The percepts can be provided any array of multi-modal sensors. The output from these sensors, a noise resistant (probabilistic) categorization of some environmental stimuli, is the basis for further processing. The perceptual categories (categorized sensor outputs) are related to the symbols exported by the symbolic system by using many ‘mini-theories’ (solutions) of how the perceptual categories should be interpreted. Each solution in the pool encodes one of these ‘mini-theories’. A genetic algorithm is used to test many of them in parallel. This results in a scalable and robust method for mapping perceptual categories to their symbolic counterparts.

The complete system can be seen as constituting a software agent. The important components and properties of the agent with respect to the framework are:

1. A symbolic system

2. An array of sensors
3. A component that realizes a genetic algorithm (including a pool of solutions that map ART-template identifiers to symbols and a list of activated templates)
4. A list of (recently) seen expressions

All the components captured by this short enumeration will be described in detail and slowly strung together. It is important to note that the existence of a concept cannot be in the functional relation between the inputs to one another. This would concern the functional role of a concept and is thus a job for the symbolic system and not for the component that maps computational percepts (the genetic algorithm).

5.2 Symbolic system

The framework should be able to accommodate any symbolic system. Keeping the number of constraints on the symbolic system as small as possible will make it easier to integrate existing symbolic systems. The symbolic system can be integrated into the proposed framework given that it satisfies the following constraints:

- It is computationally cheap to evaluate an expression. If the evaluation is unavailable or computationally too expensive, an approximation suffices as well. Note that there are two kinds of interaction with the symbolic system:
 1. The agent asks for evaluative feedback for an expression.
 2. The agent sends the definitive mapping to the symbolic system. This is the actual expression that the symbolic system needs to incorporate into its knowledge base, to use for further inference, and so on.

(1) occurs quite often and may be a cheap approximate estimate, while (2) is effectuated exactly following the operational semantics of the symbolic system.
- The functional relations in the symbolic system reflect the functional relations between external entities perceived in the world (structured combinations of percepts).
- The system can discriminate between valid and invalid expressions in a restrictive manner. This means that the number of invalid expressions clearly outweighs the number of valid ones.

The symbolic system that is applied to the toy problem is capable of determining whether an arithmetic expression is syntactically and semantically valid. It will be encapsulated by a component which maps positive or negative results (valid or invalid expressions, both syntactically and semantically ones) to some real value. See section 5.5 for an elaboration on this.

5.3 Sensors

In general any type of unsupervised clustering algorithm can be used. Fuzzy Adaptive Resonance Theory (ART2 and Fuzzy ART are used interchangeably) was selected for the concrete implementation. It is relatively easy to implement and has some interesting properties with respect to extendability and robustness. The ART-2 component in this architecture is used as an adapter for any type of sensor data. This could for example be the output of a neural network, temperature values from a heat sensor or a speech signal. The ART-2 system deals with noise-resistance and discretization (to a somewhat extreme extent) of the input and returns a steady set of templates for any input pattern. From the cognitive standpoint the ART-2 system can be seen as realizing some crude form of categorical perception. The words ‘template’ and ‘perceptual category’ will be used interchangeably.

The set of templates may be partitioned for every modality (sensor type) to prevent re-sharing of a template of a different sensor type. Each template has an unique number assigned to it ($n \in \mathbb{N}^+$ and $n = \llbracket \text{solution} \rrbracket$) that encodes the exact location of a mapping from templates to symbols on the chromosome of a solution. The template name space is shared among all the partitions.

Example:

There are two inputs (I_1, I_2) which result in two templates (a and b). We assume that they are sufficiently different from one another. Template a resonates with input I_1 and template b resonates with input I_2 . Now template a gets assigned the identifier 1 and template b the identifier 2. When a template resonates, the algorithm knows which part of the chromosome encodes knowledge specifically about that template.

In the toy problem ART-2 is used in a more direct form. It is used as a generic unsupervised classifier for sensor output. The raw images are fed directly into it with virtually no pre-processing. The bitmap images consist of byte-type pixels of gray values. These are normalized to the range $[0, 1]$ and fed to the Fuzzy-ART classifier resulting in a match (it resonates) to an existing template or a new template if none match. The presentation of the bitmap image should be seen as a specific type of optical sensor output.

All of the solutions share the same Fuzzy ART-system for categorizing perceptions during the experiments mentioned in chapter 6.

5.4 Horizon

The fitness of a solution doesn’t depend on the evaluation of a mapping considering just a single expression, but multiple expressions should be taken into account. This must be the case, because in general a single expression will probably not contain all possible inputs and will therefore not contain enough information to find the global optimum. We will call the number of expressions, on which the fitness is based, the *horizon*. The number of solutions as limited by the horizon will always start out small. This gives an early advantage, because fitness evaluation is relatively cheap. A smaller horizon will also be less

accurate in guiding the solutions in the pool towards the global optimum.

Another effect of the horizon is an implicit form of a ‘divide and conquer’ strategy [15]. In this common strategy for EA a solution is split up in multiple parts (subsolutions) and the fitness of these parts is evaluated in parallel. In the end, parts with a high fitness value are combined sometimes resulting in better solutions [40]. In our system the divide and conquer is more implicit. The number of genes that are eligible for mutation and recombination are those that are triggered by experience, all other will remain fixed. We will call these genes/parts of solutions *activated*. A more detailed treatment of the activated parts of solutions and the effects on mutation is given in section 5.6.

There are several other intuitions underlying the use of a horizon:

- The use of a horizon gives us a clear method to balance between different demands. On the one hand we want to have more information for a more accurate fitness value. On the other hand we want to have a more focused search with stringent constraints. The balance depends on the complexity of the task, inherent ambiguities between inputs, stationary/non-stationary nature of the task or computational costs of evaluating a mapping in the symbolic system. All these have an influence on how large the horizon should be. This is especially important for non-stationary tasks.
- With a smaller horizon, the search is more focused on a subset of the genes in a solution. This reduces the search space considerably.

When the number of stored inputs (only the triggered perceptual category actually needs to be stored) exceeds the size of the predetermined horizon; the oldest input gets removed from memory in a FIFO (First In First Out) manner. The accuracy of the fitness value for a solution increases with the size of the horizon, but it introduces a computational penalty as well.

5.5 Genetic algorithm

The genetic algorithm (GA) described here tries to be as simple as possible. Pointers are given to extend it to a true multi-objective setup and to add resource sharing to improve performance. These can improve performance and widen the scope of the framework. The genetic algorithm described below tries to solve the toy problem and to establish the common properties and deficiencies. The genetic algorithm is essentially a black-box allowing further (problem specific) optimizations when needed.

The genetic algorithm is actually the core of the framework. The solutions in the pool will each realize a possible interpretation of the ART template identifiers. Transcribing these to the symbols encoded in their genes will result in a symbolic expression. This symbolic expression can be judged by the symbolic system. The fitness of a mapping¹ is defined as the average fitness over a certain number

¹Seeing the fitness value as a reward signal may be insightful for those with a Reinforcement Learning [37] background

of inputs (a list of ART template identifiers). The symbolic system's main job in the fitness function (equation 5.1) is to provide a judgement of a generated expression.

$$Fitness = \frac{\sum_{t=0}^n score_t}{n} \quad (5.1)$$

In equation 5.1:

- n is the size of the horizon
- $score_t$ is 0 when the application of a mapping of the list of template identifiers for element t in the horizon is unsuccessful (not resulting in a valid expression) and is 1 otherwise.

The GA uses elitist tournament selection. As previously mentioned the proposed architecture is implicitly steady-state. Actually it depends on the size of horizon. When $||horizon|| = \infty$ or $horizon = ||inputs||$ the algorithm is no longer of the steady-state variety. This is because the full solution will then become 'active' after a certain amount of inputs or if all the perceptual categories are triggered by the elements of the horizon.

Creation of new solutions (offspring) is done by performing uniform crossover on the two parent solutions. Each pair of parent solutions will generate two new solutions. This will be done until the new solution pool has the desired size. There is an important difference between how the active and inactive parts of the parent solutions are treated by the crossover operator.

- *Active*
For each offspring uniform crossover is performed with $p = 0.5$ on the parts of the solution that are marked as active.
- *Inactive*
Each new offspring receives the exact same inactive parts of the solution from one of the parents. The rationale for this is that there is linkage in these inactive parts that is unavailable to the GA. There is no input activating those parts in the horizon, meaning that there is no fitness available to assess its value.

In each generation only certain parts of the solution are eligible for change, by being matched to the input received or alternatively; they are activated. The duration (measured in terms of queries to the symbolic system) of a generation will grow from the beginning until the number of time steps/expressions reaches the size of the horizon. It will remain constant from then on.

Mutation and selection only affect the activated categories and not the untriggered ones². This allows for retention of the mapping of rarely occurring inputs to symbols. This retention is also facilitated by using the ART-architecture, because recent inputs do not degrade previously stored information as is sometimes

²See section 5.6 for additional details

the case with neural networks. The job of the framework is to map perceptual categories to symbols, but it is allowed to map only a subset of the perceptual categories to symbols. The perceptual categories which do not map to a symbol are said to be *unmapped*.

Parts of the solution that are unmapped have no effect on the fitness value, because there was no corresponding perceptual category. This means that not mapping certain templates to symbols does not incur a penalty (fitness wise) if it has no effect on the validity of the resulting expression. On the other hand we wish to encourage the agent to map as many percepts as possible thereby enriching the symbolic representation. This process introduces an internal struggle between correctness and expressiveness between solutions in the pool.

When the interpretation of templates does not consist of a single correct mapping, but multiple conflicting ones, it would be advantageous to consider the use of multi-objective evolutionary algorithms. For the multi-objective case NSGA-II [9] and SPEA2 [44] are good candidates. This can happen when the stimuli are highly ambiguous with respect to the generated expressions fed to the symbolic system.

5.6 Protective mutation

The fitness function strongly depends on the stimuli that are presented. It is possible that, for some amount of time, no expression that triggers the corresponding part of the solution is available. This means that there is no feedback for those parts of the solution. Allowing the mutation of genes, which are not taken into account in the fitness evaluation, will eventually degrade those genes to random noise. Especially when there are long time periods between the parts of the solution and their respective activating inputs. Previously stored information will not degrade over time, because only active genes are allowed to mutate. This will decrease the number of potential beneficial mutations in the early stages of training.

At each generation the activated parts of the solutions are determined by considering the perceptual categories present in the horizon. Only a single set of such activated parts is needed, because the mapping between ART-template identifiers and their respective location on the structure of a solution is shared by all solutions in the pool for a given generation. After performing recombination on the two selected parent solutions, the resulting solution is mutated. During this mutation process all the solution parts are eligible for mutation *except* the activated parts as stored in the set.

5.7 Resource sharing

Basic genetic algorithms have the property of converging to a single solution. They will first start with an advantageous mutation by a solution in the pool that, after a number of generations, (this depends on the tournament size) becomes the dominant one in the set. The genotype will slowly move towards

this single, fitter, genotype, because it offers a higher fitness and thus a higher chance of creating offspring. This has certain drawbacks such as:

- If the solution with the highest fitness is a local optimum, other paths can be abandoned (perhaps leading to a global optimum) and we become stuck in the local optimum.
- A mono-culture adapts worse to a changing environment. With more genetic diversity in the pool a number of solutions constituting a niche can quickly grow to become the dominant solution. This can happen due to changing stimuli or modifications to the symbolic system.

We want to motivate other solutions to consider alternatives that may not yet be good, but will become so in a few generations. We don't need to have the whole population to converge to the same optimum, because only a single solution is enough to work with. Elitism makes sure that the optima found thus far does not get lost.

The following algorithm implements a very rough, but effective, type of resource sharing that reduces the acquired fitness if multiple solutions score (generate a correct expression) on the same percept. For example: if the global optimum (O) scores positive on 8 out of the 10 expressions, normally O will have the highest chance of survival. Solutions that target niches on which O does not score will disappear over time, because the solution pool will slowly become a mono-culture. To counter this, the remaining ones, targeting a niche, will be stimulated to score on the remaining expressions. They will have an elevated fitness on success, ensuring survival when compared to the offspring of O which will probably target the same percepts. This happens even though they score on fewer expressions.

Listing 5.1: Fitness modification

```
1 solutions.sort!() #by unadjusted fitness value
2 percepts.resetcounters()
3 solutions.each do |solution|
4
5     solution.fitness =  $\frac{\sum_i^{||percepts||} \gamma^{counter_i}}{||percepts||}$ 
6
7     solution.getActivatedPercepts().each do |percept|
8         percept.increment_counter()
9     end
10 end
```

The decay factor γ is a real number in the range $[0, 1]$. This sets the rate at which the value for a correct mapping of an expression drops. The lower the value the quicker the value drops and thus the smaller the number of solutions that can profit from getting the mapping for that list of templates correct.

5.8 Runtime

This section will cover the various phases in the framework starting from the presentation of an input up to the selection of the final symbolic expression by the genetic algorithm. The same overview is given in pseudocode in listing 5.2.

Before the very first exposure of an expression to the system, the solution pool is initialized with random solutions (random mappings from ART-templates to symbols). The fitness of a solution is evaluated by the symbolic system resulting in a fitness value. The complete list of all solutions, representing all of the possible interpretations of the perceptual categories, is called the *solution pool*.

The system begins with the processing of an input. These inputs consist of random correct algorithmic expressions which in turn consist of bitmap image sequences. These are fed to the Fuzzy ART system resulting in a categorization (perceptual category) of the input. The Fuzzy ART system classifies the images one-by-one instead of the whole expression in one go. This is because the templates are created at the level of the individual symbols and not at the level of an expression. After all of the images have been mapped to an ART template identifier, all the solutions in the pool are used to interpret the template identifiers and generate symbolic expressions. The feedback from the symbolic system about the generated symbolic expressions determines the fitness value of the solutions. The average of these fitness values then determines the chance of creating new offspring in combination with another solution (using uniform crossover).

The ongoing process of acquiring inputs, evaluating individuals, applying selection, recombination and mutation continues for the number of set generations and available number of inputs (possibly infinite). An agent, in the end or during exposure to inputs, can have a set of (partially) incompatible solutions to solve the task. This allows for context ambiguity where the presence of certain inputs changes the interpretation of other inputs. The intuition for this comes from the idea of having ‘multiple drafts of consciousness’ all competing for resources and availability. This set of solutions is sorted and the one with the highest fitness is triggered first. If it does not result in a positive response from the symbolic system, a second one is tried, and so on.

Listing 5.2: The full proposed framework in pseudo-code

```

1 pool = Pool.new(n, random)
2 horizon = Horizon.new(max_size)
3
4 generations.times do
5
6   newpool = Pool.new(0)
7
8   horizon += ART2.get_template_list( Environment.getInput() )
9
10  #interpret each template in the lists as a symbol and
11  #evaluate the resulting symbolic expression
12  pool.each do |solution|
13    solution.getFitness(horizon, Symbolic_system)
14  end
15
16  #elitism
17  newpool.add(pool.getBest())
18
19  #resource sharing
20  pool.sort.each do |solution|
21    solution.recalculate_fitness_applying_resource_sharing()
22  end
23
24  #apply natural selection using binary tournaments
25  while ( newpool.size < pool.size ) do
26    solution1 = pool.get_solution()
27    solution2 = pool.get_solution()
28
29    newsolution = Solution.recombine(solution1, solution2)
30    newsolution.mutate()
31    newpool.add(newsolution)
32  end
33  pool = newpool
34 end

```

When a new computational percept (a sensor) is added to the system, the existing structure of every solution in the pool needs to be modified. The size of the solution will increase with the number of new computational percepts and the number of symbols on which to map them. The new solution-part is added and randomly initialized. This addition has no effect on the existing solution structure, thereby allowing incremental learning with no deteriorating effect on previously acquired knowledge.

Assuming that the population has converged to the global optimum on the previous inputs and the possible states of the symbolic system, the general structure of solutions in the pool will presumably be fairly uniform. In the multi-objective case there will be multiple uniform groups. When adding a new sensor to the symbolic system, we are in fact enlarging the existing solution structure of every solution in the pool. This new addition (the new sensor) then becomes the only way to get competitive advantage over other solutions in the pool. This only holds when assuming that the solution pool consists of fairly uniform solutions that represent the global optimum. In turn this will

lead to a focused search for the correct mapping of the new percepts to an existing symbol. The system is expected to achieve the new global optimum rather quickly.

The average fitness is used to monitor performance over extended periods of time. If for example the performance of the system is not satisfactory, the vigilance parameter (ρ) of the ART-system could be increased to make the computational percepts more fine grained. This allows the agent to discriminate between certain concepts that were previously lumped together. There is a downside in terms of expansion of the search space when increasing the number of computational percepts.

Chapter 6

Results

This chapter will give an overview of the results for the different experiments of the implemented framework that have been run. The main objective is to establish common patterns that hold for arbitrary tasks. This is attempted by relating and explaining the various differences in performance for different problem sizes and parameter settings in a generic way.

The experiments mentioned henceforth were run with the following parameters unless mentioned otherwise:

- Solution pool size = 2000
- Tournament size = 3
- Mutation rate = 0.02
- Protective mutation = true
- γ (Resource sharing decay factor) = 0.95
- The default problem set consists of expressions of the form: “4+7=11” using the operators ‘+’ and ‘=’.

All the results are averaged over 50 runs only considering the best performing solution in that generation (the one selected by elitism). Template identifiers were directly fed to the genetic algorithm for performance reasons instead of performing the Fuzzy ART classification. Experiments were run with multiple template identifiers realizing the same symbol to account for sub-optimal classification.

6.1 Fixed horizon

All experiments in this section are run on a fixed horizon. This means that the horizon doesn’t change during the entire run. The horizon does consist though of different randomly generated expressions for every run.

These runs with a fixed horizon can be seen as indicators for using the algorithm in a special case of batch learning (here only 1 batch). Additionally they are used for estimating how much information a certain horizon size contains on average. These results can then be used to select the optimal horizon size if time is of the essence.

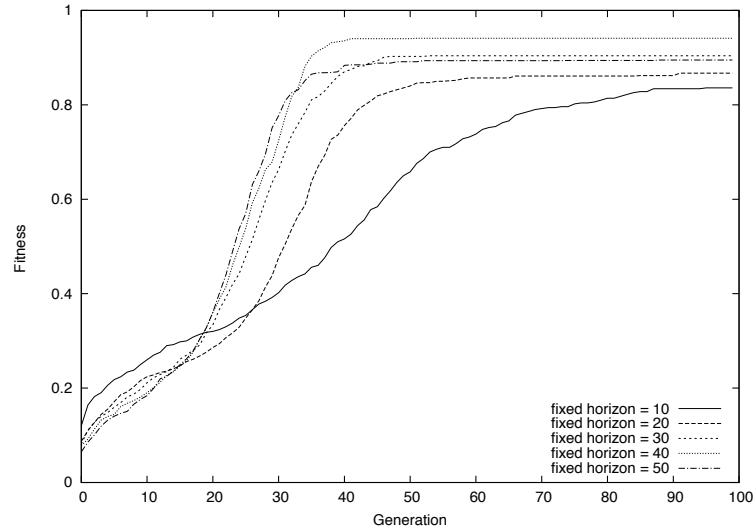


Figure 6.1: Comparison of different fixed horizon sizes for a single perceptual category per symbol

Figure 6.1 clearly illustrates the advantages of increasing the size of the horizon. There is an improvement illustrated by the steepness of the fitness curve. It appears to be the case that increasing the horizon only helps up to a certain extent, because the improvement after a horizon size of 30 is negligible. This seems to suggest that the increased horizon sizes do not offer any additional information that leads to a higher fitness when compared to the smaller horizon sizes.

Results are quite similar in figure 6.2 when compared to figure 6.1. In figure 6.2 we have two percepts for each symbol. A clear improvement is again seen when increasing the size of the horizon.

The case for 5 templates for each symbol, in figure 6.3, looks to be quite a bit more difficult than the previous experiments. We see that the slow positive trend continues even after a 100 generations, a lot longer than the one and two templates per symbol tasks.

We see that performance stabilizes after about 150 generations. The slower improvement in performance and larger number of generations seems to indicate that this problem is harder (and rightfully so). There doesn't seem to be much of a difference between the different horizon sizes except the smallest one (40).

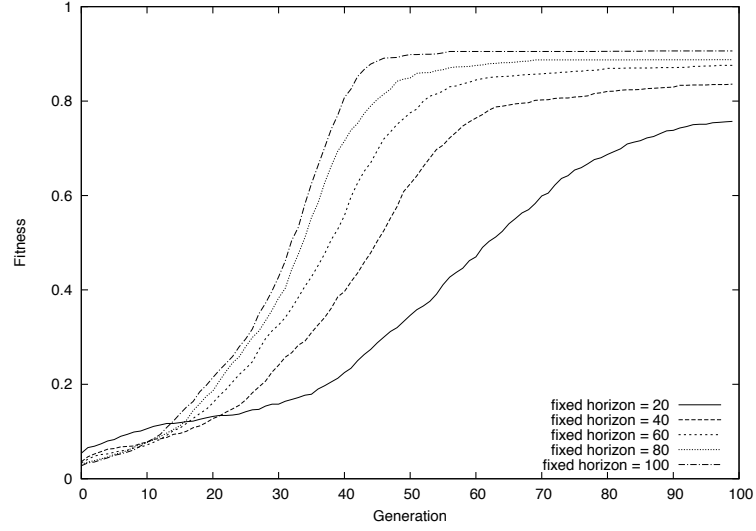


Figure 6.2: Comparison of different fixed horizon sizes for two perceptual categories per symbol

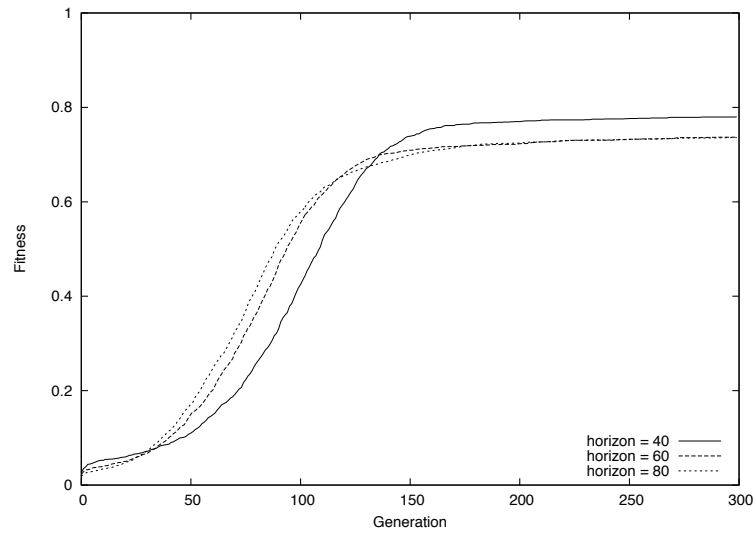


Figure 6.3: Comparison of different fixed horizon sizes for 5 perceptual categories per symbol

The significantly better performance on the “horizon = 40”-case is intriguing and perhaps at first in conflict with the main idea behind the relation between the fitness and the size of the horizon. A larger horizon is said to increase the quality of the fitness making it smoother and better suited at guiding the solution pool to a global optimum. The increase in horizon size also has an influence on how difficult the task is. A smaller horizon allows the system to achieve a high fitness (let’s say 1.0), but it is possible that it is still (objectively) a local optimum. Each expression in the horizon constrains the possible interpretations of the percepts that yield a high fitness. Therefore a smaller horizon has fewer constraints, making the task easier at the cost of decreasing the quality of the fitness with regard to the global optimum. There is a turning point where the increased quality of the fitness value (an increased horizon) outweighs the difficulty of the task (expressed in the restrictions on the interpretation that each expression in the horizon introduces).

Turning back to figure 6.3 we expect that the smaller horizon sizes also lead to improved performance, but at the cost of a larger chance of being stuck in a local optimum instead of a global one.

Performance on the more difficult task using five templates for each symbol still isn’t up to par with the performance achieved on one and two templates per symbol. We expect that this is due to not having enough building blocks available. An increase in population size is expected to improve performance. The flipside is of course a significantly higher computational load.

The question that rises with a fixed horizon is: when do you know that the low fitness value is due to a horizon that is too small? Is increasing the horizon the only thing that you can do to improve the achieved fitness?

6.2 Dynamic horizon

A dynamic horizon has a fixed maximum size just like a fixed horizon, but its content is not static. A new expression is added to the horizon every generation which causes the oldest element to be removed when the dynamic horizon reaches its maximum capacity. The dynamic horizon stores the last N presented inputs as a kind of short-term memory.

A dynamic horizon should yield comparable performance to that of a large fixed horizon. At least when letting it run for a sufficient number of generations. This is supported by using protective mutation which has a stronger effect when applied to more perceptual categories (ART-2 templates) and the resulting larger solution. Furthermore, the fitness function doesn’t go up in a strict hill climbing fashion, but will drop from time to time even when employing elitism. This is caused by the changing fitness function. It changes, because the fitness is based on the changing list of expressions in the dynamic horizon.

We expect that using a dynamic horizon will perform worse (achieve a lower fitness value) than when using a fixed horizon, but that the final mapping will be closer to the true global optimum (because a larger part of the total input space can be sampled). We also expect that it is cheaper computationally wise (it needs less fitness evaluations) when compared to the fixed horizon.

Figure 6.4 shows the difference between a fixed and dynamic horizon and shows that a smaller dynamic horizon can achieve the same or a higher fitness as a larger fixed horizon.

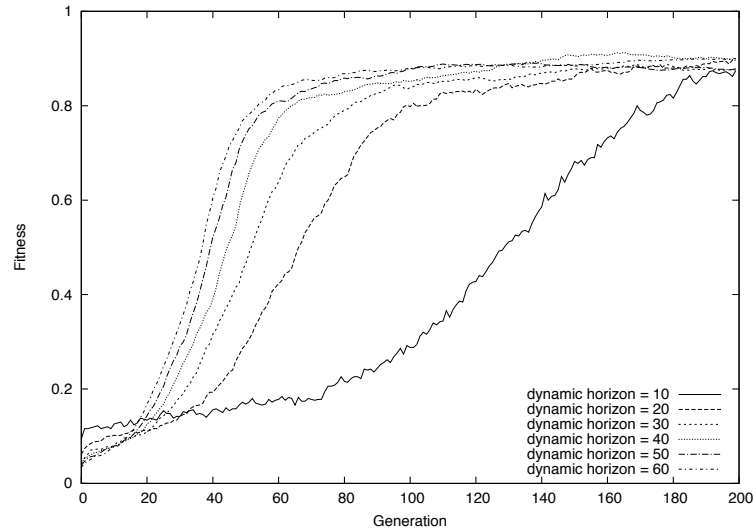


Figure 6.4: Performance on two templates per symbol with different dynamic horizon sizes

6.3 Performance

The computational load is largely determined by the evaluations provided by the symbolic system, since these have to be given for all the solutions in the pool on all the items in the horizon. The worst case number of evaluations is then at generation g :

$$evaluations = ||solutions|| \cdot ||horizon|| \cdot g \quad (6.1)$$

Usually g is the number of generations where the fitness value stabilizes. This doesn't account for the quality of the mapping found though so it could be modified to incorporate the achieved fitness value.

In order to determine if the dynamic horizon-approach is actually more efficient in attaining the global optimum some measure of *objective fitness* is needed. This objective fitness is an indication of the selected solution's distance to the global optimum. The lower the score, the further it is from the global optimum. The objective fitness is calculated by filling an additional, previously unseen,

set with expressions (realized by multiple percepts). The solution's fitness is calculated by taking the average fitness obtained from evaluating each output of the solution for an expression from this set by querying the symbolic system. The size for this 'test set' is 100 unless indicated otherwise.

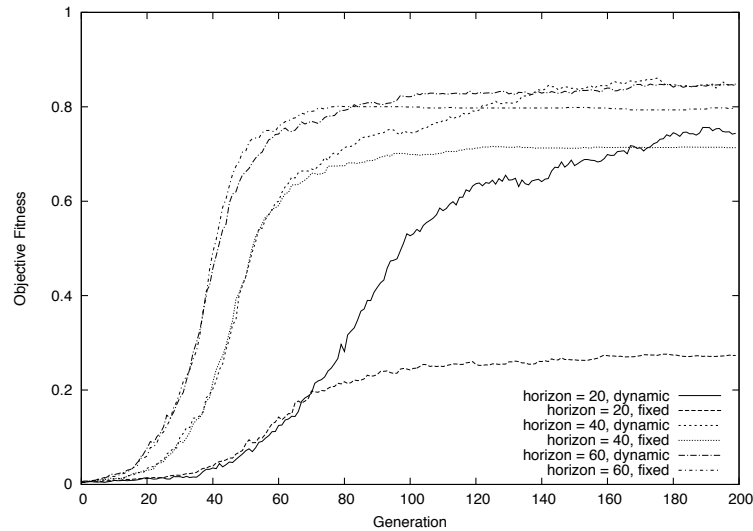


Figure 6.5: Comparison between the objective fitness for a differently sized fixed and dynamic horizons with two templates for each symbol

Figure 6.5 nicely illustrates the expected differences between a dynamic and fixed horizon. The same pattern occurs with all the horizon sizes. The dynamic and fixed horizon overlap until the information in the fixed horizon has been maximally exploited. After that, in the case of the dynamic horizon, the fitness value continues to rise, because more inputs are sampled than in the fixed horizon. There really doesn't seem to be a downside to using dynamic horizons. In the worst case the dynamic horizon yields the same fitness value as a fixed horizon if its size is sufficient to capture all of the relations present in the inputs.

6.4 Protective mutation

It was hypothesized that protective mutation would lead to better performance, because of prevented mutation of unactivated parts of the solution and the (possible) linkage kept by unaltered copying from one parent solution during recombination. The graph here shows the performance difference between two runs, one with protective mutation and one without.

We can see that in figure 6.6 protective mutation outperforms (comparing the highest objective fitness at the last generation) the unprotected variant for a

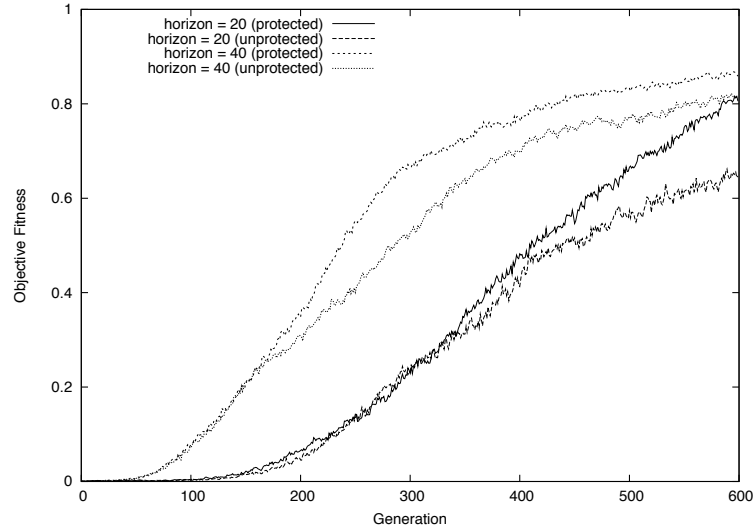


Figure 6.6: Comparison of runs with and without protective mutation using a dynamic horizon using 5 templates for each symbol

horizon size of 20 ($p = 0.004$), but not statistically significant for a horizon size 40 ($p = 0.219$). The “horizon = 40”-case difference is less strong, because the protective mutation in that case offers less of a performance benefit, because a larger part of the solution will be activated. Since the majority of the solution parts is activated, there is fitness feedback for them. This means that the quality of the pool actually increases, because newly generated solutions are given their respective higher fitness value when generated. In the “horizon = 20”-case, less of the solution is activated and protective mutation actually preserves information, because there is no fitness feedback for the complete solution. Mutating parts of the solution that are not activated would decrease the quality, because there is no fitness feedback for those parts to guide the selection process.

6.5 Adding a sensor

Adding a sensor to a system that has already converged to the global optimum should be trivial, because the population will largely resemble a mono-culture. The only way to increase the fitness for a particular solution is by learning the correct mapping for the new sensor. This behaviour is further encouraged by using resource sharing. Resource sharing will cause a major increase in fitness for the few solutions that find a correct mapping for the new templates when compared to the rest of the pool.

Adding a new sensor for an already existing symbol is the same as introducing

a new template for an existing concept. The length of every solution will have to be increased to allow for an additional template to symbol mapping.

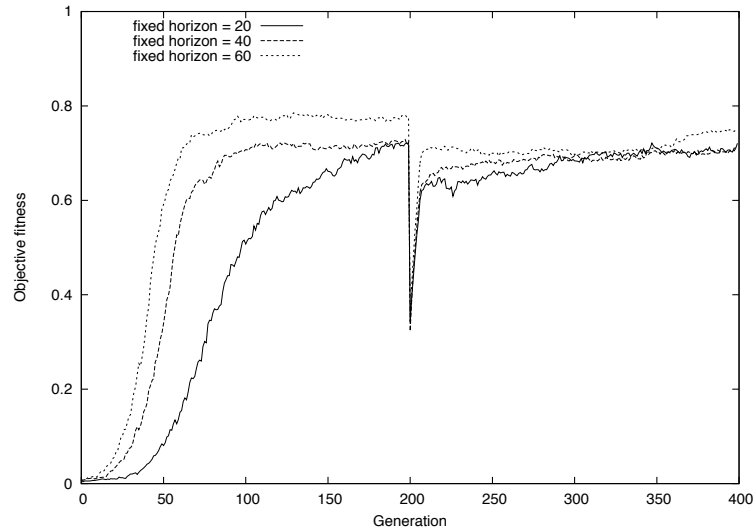


Figure 6.7: Comparison of performance on different horizon sizes for two perceptual categories per symbol. After 200 generations four new unseen templates realizing two new symbols are added to the inputs and are introduced in the objective set in a big bang.

Figure 6.7 shows the objective fitness value when adding symbols in a big bang fashion on a trained system. Where *big bang* means incorporating the newly added symbols directly in the full horizon radically changing the fitness value for the current solutions.

An example of when this situation (adding the symbols in a big bang fashion) could occur:

The system is trained in a certain environment and finds an effective mapping of computational percepts to symbolic expressions. At night, the system is turned off and moved to a different environment. The system then needs to relearn how to interpret the previously unencountered percepts in the new environment by maximally exploiting its old knowledge where possible.

Because the fitness function is radically different when compared to the first few generations and the population isn't as diverse as in the beginning it is more prone to converging to a local optimum.

Figure 6.8 illustrates the situation where new sensors are introduced slowly into the horizon starting from the 200th generation. The drop in performance is due to the testset changing to incorporate the new symbols and their distribution in a *big bang*-fashion. It is clear that the system quickly learns the correct interpretation for the newly introduced templates and retains the previously

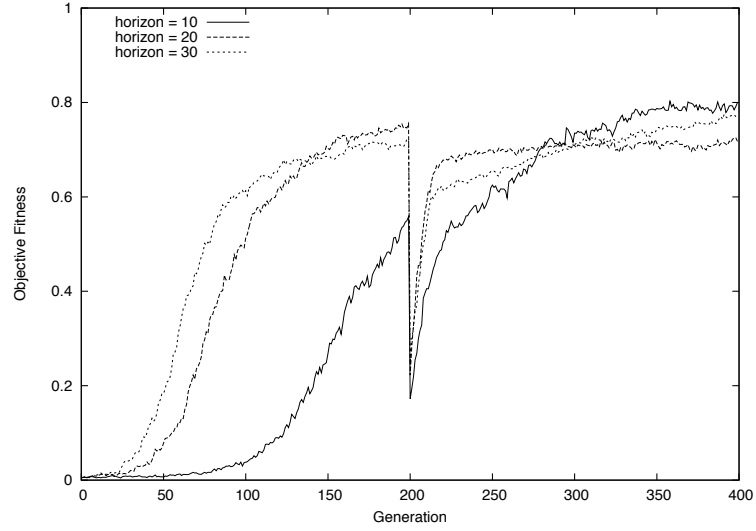


Figure 6.8: Number of generations needed to acquire 4 new templates (previously unseen symbols) after being trained for 200 generations, slowly adding them to the horizon at a rate of no more than 1 per generation.

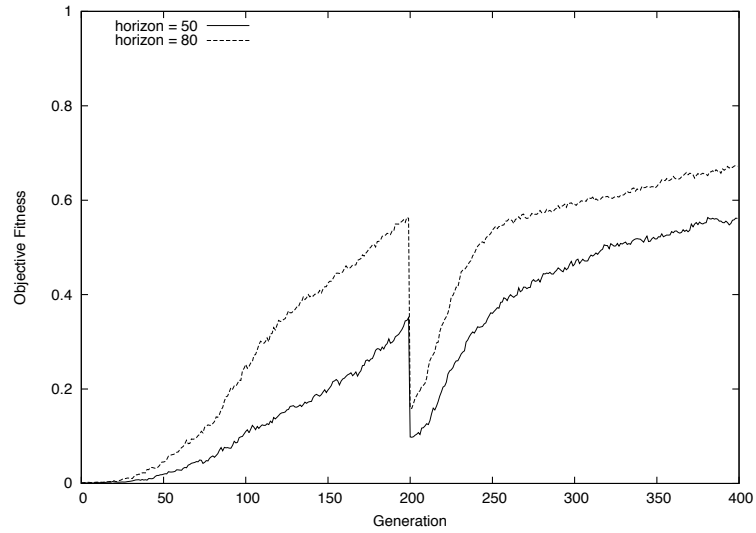


Figure 6.9: Number of generations needed to acquire 10 new templates (previously unseen symbols) after being trained for 200 generations, slowly adding them to the horizon at a rate of no more than 1 per generation.

acquired knowledge. It even seems to have only a minimally disruptive effect on the learning performance before convergence when looking at the “horizon=10”-line. Figure 6.9 shows a similar, but harder setup with 10 new templates.

6.6 Learning a new symbol

Figure 6.10 shows the average fitness trend when adding a new symbol to the symbolic system after 200 generations. It starts with all the digits and the operators ‘+’ and ‘=’. After 200 generations the operator ‘*’ is added. It starts off worse when compared to the other experiments, because the fitness value is severely impoverished. This is due to large parts of the horizon containing sensor values for which no correct mapping is possible (percepts realizing the ‘*’ operator). We see however that the fitness increases quickly to incorporate the availability of the new symbol after adding it to the symbolic system. The difference between a horizon of 20 and 40 is small though significant ($p=0.00060$). We do see however an increase in performance when decreasing γ from 0.95 to 0.90 for the “horizon=20”-case ($p=0.00002$). The “horizon=40”-case profits less from this, because it loses more of its building blocks in the first 200 generations. The “horizon=20”-case is more affected by setting γ to 0.90, because there isn’t much diversity left in the pool in the “horizon=40”-case because of the larger horizon. ($p=0.01224$)

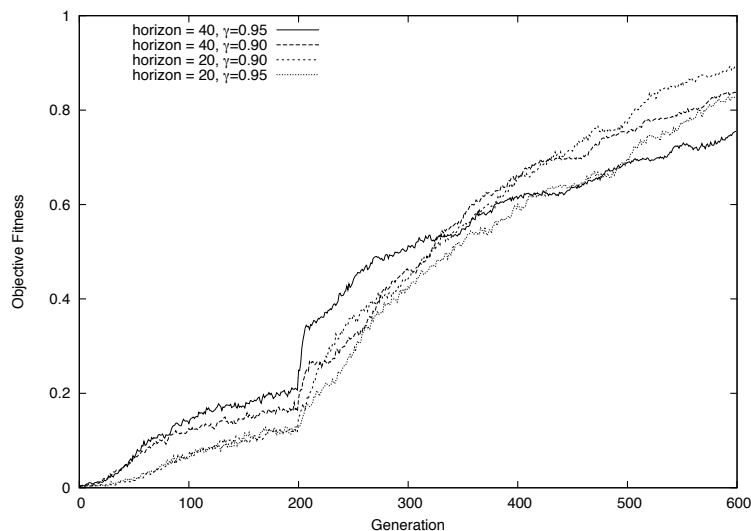


Figure 6.10: Number of generations needed to correctly map a new symbol that was previously unavailable in the symbolic system, but was available as a sensor output. The new symbol is added after 200 generations in the symbolic system. Two percepts for each symbol were used.

Chapter 7

Conclusion & Discussion

7.1 Conclusion

7.1.1 Agent engineering perspective

Is it possible to align arbitrary sensor data with some symbolic description? And if so, what are the limitations and preconditions?

It has been shown that the relation between symbols and connectionist sensor outputs is learnable in an incremental, scalable and robust manner. Nothing is assumed about the relation in advance, but it follows from the functional relations between the symbols in the symbolic system. The feedback from the symbolic system shapes the relation and brings down the semantic barrier between the two.

Fixed horizons can be used, but dynamic horizons offer compelling advantages. Dynamic horizons converge about as fast as fixed horizons on sizes that approach the minimal characterizing set as illustrated in section 6.2. When a dynamic horizon is used, similar performance (here meant as the highest fitness achieved after a number of generations) can be attained with a smaller horizon resulting in lower computational costs and less risk of premature convergence. This also allows for learning larger domains where certain templates may be absent from the input for an undefined period of time due to protected mutation. The results from section 6.5 illustrate that new sensors can be added without degrading previously acquired knowledge. Section 6.6 shows that new symbols can be added incrementally thereby, supporting incremental learning and the automatic exploitation of new symbolic knowledge. New symbols or sensors are incorporated quite quickly in existing solutions. In the default setting the relation between symbols and sensors is manually defined. The automatic integration and interpretation of new sensors and symbols is something that is not possible in the default setting. There are some limits with respect to the search space of all possible mappings when starting, but this can be circumvented by incrementally adding symbols and or sensors on the fly.

Given the amount of attention paid to creating a generic approach, the framework described here can be used to combine existing symbolic systems with

arrays of existing sensors. The effort involved in linking the two can be reduced considerably, because symbolic systems and generic sensors are a lot easier to develop and maintain separately than to manually create and maintain a hybrid system. The framework is capable of automatically learning the mapping from sensors to symbols. This allows existing symbolic and connectionist systems to be linked and deployed which automatically improve when adding better sensors or improving the symbolic structures. This offers great advantages from a software/agent engineering standpoint, allowing further decoupling of the symbolic and connectionist components. The framework furthermore assumes very little about whether the task is stationary, ambiguous or noisy and allows for automatic optimization with respect to the specific conditions of the task.

7.1.2 Philosophical perspective

Is the functional/inferential role of a concept sufficient to acquire the “ability to categorize”? What needs to be assumed about such a process to be a viable explanation of an agent that grounds concepts in its knowledge base in accordance with a hybrid architecture?

In the experiments there is no question about “what keeps the two together” [10], because the system could effectively acquire the relation between the two in the experiments that were performed using the functional (inferential) role of the concept embedded in the symbolic system. This without any assumptions about the relation in advance, because the solutions in the pool were initialized with random values. The successful experiments further strengthen the arguments supporting a two-factor CRS by showing that the toy problem is solvable. The components of the framework are mapped to their philosophical counterparts. It has been argued that extreme nativism and making a distinction between categorization and identification successfully captures the proposed hybrid framework. This atomic view on concepts, entailed by extreme nativism, is a generally accepted position in symbolic AI and or formal logic with respect to artificial systems.

The results from the specific implementation strengthen the arguments concerning the interaction between the narrow and wide semantics in two-factor CRS. The relation between wide scope semantics and narrow scope semantics can evolve automatically, assuming that they are sufficiently alike in the sense that they realize about the same concepts only by different means. Symbolic systems realize atomic concepts with the conceptual role of a symbol. Connectionist systems realize abilities to categorize stimuli by means of their structure.

Two-factor Conceptual Role Semantics thus seems to be an excellent philosophical framework that captures these two relations between symbolic and connectionist systems in this type of hybrid AI systems from a theoretical philosophical perspective. CRS allows us to link both the narrow and wide semantics to their respective artificial components (connectionist and symbolic systems). The question of whether the narrow semantics can evolve into a method for identifying objects external to the agents remains a direction for further research. Especially when the two presuppose different conceptualizations or different levels of abstractions. The thing that keeps the two together in the framework is

natural selection applied to the pool of possible semantic interpretations (solutions mapping perceptual categories to symbols). This all while assuming some form of extreme nativism (as realized by the symbolic system).

The combination of nativism with CRS links the philosophical enterprise to the research in knowledge based hybrid AI systems and could lead to fruitful collaborations in the spirit of the language and concept evolution / acquisition / alignment research by Steels [35].

7.2 Discussion

Even though the preliminary results are fairly positive with respect to this approach several questions remain:

1. Even though experiments for one, two and five perceptual categories for each symbol have been run it is not at all clear if the trend for incremental learning will continue. Already there's a small drop in average performance when increasing the number of symbols (probably due to the algorithm getting stuck in more local minima.). The question if this approach can deal with more complex problems is not answered in this thesis.
2. The results for the symbol adding experiments in section 6.6 show that the system is able to map previously acquired sensor outputs to newly added symbols. The performance on the previous 200 generations is very low though, when looking at the objective fitness measure. This seems to suggest that the system is not very well suited for domains where this mismatch between sensor outputs and lack of symbols is more common. In turn, this would mean that the system is more prone to getting stuck in local optima. Some of this mismatch is captured by the requirement that the symbolic system and the sensor outputs share their level of conceptualisation, but this result hints at a requirement that is somewhat stronger than that.
3. When trying to add two operators at once ('-' and '*') with two percepts for each symbol while sensor outputs for those percepts already exist, the system gets stuck in a local optimum as is shown in figure 7.1. The fitness value is so impoverished (roughly two-thirds cannot be mapped when the symbols are unavailable in the symbolic system) that the system converges to a local optimum and the initial pool diversity gets lost. It seems that the fitness value needs to be of sufficient quality before adding additional symbols. The quality is important if the system is meant to be able to support incremental learning.

7.3 Further research

There are many interesting paths with respect to this problem which could have been taken that target fascinating questions. Next follows a non exhaustive list of such research directions that one could take:

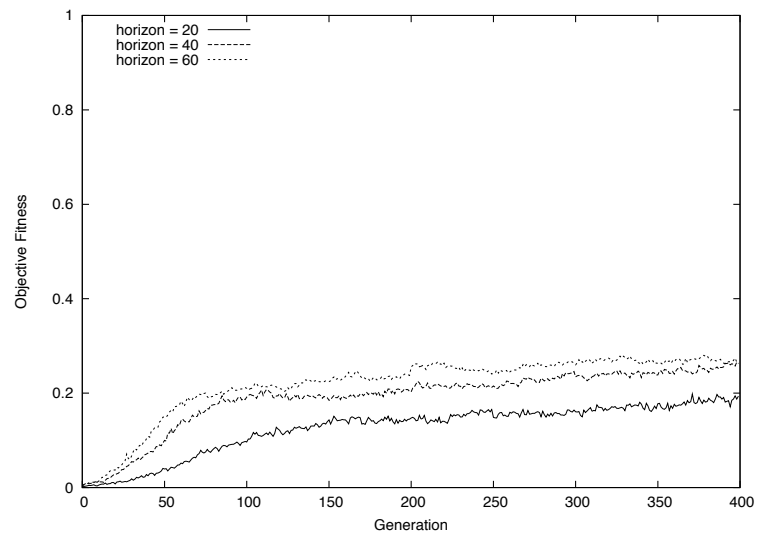


Figure 7.1: The system is trained on the usual set with the operators '-', '+', '*' but only the '+' operator is available in the symbolic system for the first 200 generations. After 200 generations the remaining operators are added and the system is only then able to correctly map the templates to symbols. Two templates per symbol were used.

- The functional relations in the perceived external entities as provided by perceptual categories should be captured by the symbolic system. This only works though in a situation where the connectionist and symbolic system are on the same level of abstraction (meaning that there is a many-to-one mapping from perceptual categories to symbols). A combination of perceptual categories together realizing some abstract symbol cannot be captured in the current system without changing the symbolic system. How can functional dependencies outside of the symbolic system be captured between perceptual categories in a fully automatic way? Should some kind of clustering be used, a type of higher order perceptual discretisation?
- The current framework implementation uses a fairly standard genetic algorithm while more optimized versions for these types of problems exist [9]. Do they offer significant advantages when compared to the stock one? Can ambiguities be better handled by multi-objective versions? How do you incorporate niching in order to keep solutions with a specific advantage in the pool (resource sharing hints at a possible approach but is fairly crude) even though their average fitness is lower?
- Another optimization of the fitness value is possible for symbolic systems with an ontology or otherwise hierarchical representation of knowledge. When assuming that the system encodes for general types leading to more specific ones the fitness could increase with the specificity of the resulting symbol. A very specific symbol is relatively high risk (the probability that the mapping is incorrect is much larger than for a generic symbol higher up in the tree), but gives a higher fitness. This in contrast to a generic symbol that is bound to occur more often, but will not result in much of a fitness increase. Such a setup would allow for an automatic balancing between correctness and specificity when using a symbolic system with an ontology.
- If a solution generates an incorrect expression that the agent attempted to use not purely as an evaluation, but as a definitive one, the system could enter a special routine which tries to select the best alternative solution. It makes little sense to select a solution which is very similar to the one that apparently failed, so we need a way to order them on the basis of similarity. Ideally we want to have a solution that scores on different percepts than the one that failed. An even better proposal is to identify the clusters (commonly referred to as *sub-populations*) that each constitute a typical solution (for example the center of the cluster) that together score on the full horizon. The calculation of this set of clusters sadly is equivalent to the ‘Set Cover Problem’ which is NP-complete [25].

The positive results from the toy problem and the fact that there are still many points of improvement possible that haven’t yet been explored, seem to suggest a bright future for this type of framework in hybrid AI systems. It is my hope that this work can contribute to better AI engineering practices.

Bibliography

- [1] Neural-symbolic systems. *Journal of Applied Logic*, 2:241379, 2004.
- [2] Albert Atkin. Peirce's theory of signs. <http://plato.stanford.edu/entries/peirce-semiotics/>, 2006.
- [3] Ned Block. "Conceptual role semantics" by ned block. <http://www.nyu.edu/gsas/dept/philo/faculty/block/papers/ConceptualRoleSemantics.html>, 2007.
- [4] G. Carpenter, S. Grossberg, and D. B. Rosen. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4(6):759–771, 1991.
- [5] G. A. Carpenter, S. Grossberg, Boston University, Dept. of Cognitive, and Neural Systems. *Adaptive Resonance Theory*. Boston University, Center for Adaptive Systems and Dept. of Cognitive and Neural Systems, 1994.
- [6] D. Cliff and J. Noble. Knowledge-based vision and simple visual machines. <http://eprints.whiterose.ac.uk/1277/>, August 1997.
- [7] C. A. C. Coello. A comprehensive survey of Evolutionary-Based multi-objective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.
- [8] Daniel Crevier and Richard Lepage. Knowledge-Based image understanding systems: A survey*1. *Computer Vision and Image Understanding*, 67(2):161–185, August 1997.
- [9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.
- [10] Jerry A. Fodor, Ernest LePore, and Ernest LePore. *Holism*. 1992.
- [11] Stevan Harnad. Grounding symbols in the analog world with neural nets. *Think (Special Issue on "Connectionism versus Symbolism" D.M.W. Powers and P.A. Flach, eds.)*, 2:41–43, 1993.
- [12] Stevan Harnad. The symbol grounding problem. <http://arxiv.org/abs/cs/9906002>, 1999.
- [13] Stevan Harnad, Claire Lefebvre, and Henri Cohen. *To Cognize is to Categorize: Cognition is Categorization*. Elsevier, December 2005. UQaM Summer Institute in Cognitive Sciences on Categorization. 30 June - 11 July 2003 <http://www.unites.uqam.cccog/liens/program.html>.
- [14] Christoph Herrmann. *Fuzzy Logic as interfacing technique in hybrid AI-systems*, pages 69–80. 1997.
- [15] S. Y. Ho, L. S. Shu, and J. H. Chen. Intelligent evolutionary algorithms for large parameter optimization problems. *Evolutionary Computation, IEEE Transactions on*, 8(6):522–541, 2004.

- [16] Michael Huemer. Sense-Data. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. 2009.
- [17] F. C. Keil and R. A. Wilson. *The MIT Encyclopedia of the Cognitive Sciences (MITECS) MIT Cognet library*. MIT press, 2001.
- [18] T Kohonen and T Honkela. Kohonen network. *Scholarpedia*, 2(1):1568, 2007.
- [19] E. Kolman and M. Margaliot. Are artificial neural networks white boxes? *Neural Networks, IEEE Transactions on*, 16(4):844–852, 2005.
- [20] J. E. Laird, P. S. Rosenbloom, and A. Newell. Chunking in soar: The anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.
- [21] S. Laurence and E. Margolis. Concepts and cognitive science. *Concepts: Core Readings*, page 381, 1999.
- [22] S. Laurence and E. Margolis. Radical concept nativism. *Cognition*, 86(1):2555, November 2002.
- [23] Y. LeCun and C. Cortes. The MNIST database of handwritten digits. *NEC Research Institute*, <http://yann.lecun.com/exdb/mnist/index.html>.
- [24] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, and P. Simard. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 276, 1995.
- [25] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. In *Proceedings of the 25th Annual Acm Symposium on the Theory of Computing*, page 286. Association for Computing Machinery (ACM), 1993.
- [26] E. Margolis. How to acquire a concept. *Mind & Language*, 13(3):347–369, 1998.
- [27] A. Blondin Masse, G. Chicoisne, Y. Gargouri, S. Harnad, O. Picard, and O. Marcotte. How is meaning grounded in dictionary definitions? *0806.3710*, June 2008.
- [28] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira. An introduction to the syntax and content of cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, page 4449, 2006.
- [29] Kenneth McGarry, Stefan Wermter, and John MacIntyre. Hybrid neural systems: From simple coupling to fully integrated neural networks. 2:62–93, 1999.
- [30] R. G. Millikan. A common structure for concepts of individuals, stuffs, and real kinds: More mama, more milk, and more mouse. *Behavioral and Brain Sciences*, 21(01):55–65, 1998.

- [31] H. Putnam. Meaning and reference. *The Journal of Philosophy*, pages 699–711, 1973.
- [32] D. Roy. Grounding words in perception and action: computational insights. *Trends in Cognitive Sciences*, 9(8):389396, 2005.
- [33] Elena Sapojnikova. *ART-based Fuzzy Classifiers*. 2004.
- [34] J. Searle. Minds, brains, and programs, with commentaries by other authors and searle’s reply. *The Behavioural and Brain Sciences*, 3:417–457, 1980.
- [35] L. Steels. The symbol grounding problem has been solved. so what’s next? In M. de Vega, editor, *Symbols and Embodiment: Debates on Meaning and Cognition*. Oxford University Press, Oxford, 2008.
- [36] L. Steels, M. De Vega, G. Glennberg, and G. Graesser. *The symbol grounding problem is solved, so what’s next?* Academic Press, New Haven, 2007.
- [37] R.S Sutton and A.G. Barto. *Reinforcement Learning*. MIT Press, 5, illustrated edition, 1998.
- [38] Hendrik Svensson. Notions of embodiment in cognitive science. <http://his.diva-portal.org/smash/record.jsf?pid=diva2:2975>, 2008.
- [39] D. Thierens. Scalability problems of simple genetic algorithms. *Evolutionary Computation*, 7(4):331–352, 1999.
- [40] Christine L. Valenzuela and Antonia J. Jones. Evolutionary divide and conquer (i): A novel genetic approach to the tsp. *Evol. Comput.*, 1(4):313–333, 1993.
- [41] Daniel Whiting. Conceptual role semantics [Internet encyclopedia of philosophy]. <http://www.iep.utm.edu/c/conc-rol.htm>, 2007.
- [42] A. Wilson and Hendler J. Linking symbolic and subsymbolic computing. *Connection science*, 5(3-4):395–414, 1993.
- [43] Samuel Wintermute. The SVS spatial visual reasoning system, May 2008.
- [44] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: improving the strength pareto evolutionary algorithm. In *EUROGEN*, page 95100, 2001.