The role of metaphor and embodiment in the development of mathematical concepts: A computational approach

Alexander Karl Svanevik

Master of Science

Artificial Intelligence

School of Informatics

University of Edinburgh

2010

Abstract

Lakoff and Núñez (2000) have claimed that the first mathematical concepts arise from a set of *grounding metaphors* - metaphors which ground abstract concepts in bodily experience. This project takes a computational approach to this hypothesis. We use the framework of heuristic-driven theory projection (HDTP) and demonstrate that an agent interacting with two simple environments can develop a basic conceptual apparatus for mathematics, given a set of sensorimotoric primitives, an associative memory and a capacity for forming metaphors. The result is set of association structures representing embodied mathematical concepts.

Acknowledgements

First of all, I would like to thank my excellent supervisor Alison Pease, who has been a more helpful supervisor than I could ever have wished for. Her enthusiasm for the field of mathematical cognition made the project seem worthwile even when my motivation was at its lowest. I would also like to thank Subramanian Ramamoorthy and Alan Smaill for their guidance which led to some of the main ideas in this work. Finally, I thank Martin Schmidt at the University of Osnabrück for letting me use his implementation of HDTP, and for his availability and efficiency in dealing with issues I encountered using this system.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Alexander Karl Svanevik

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The first mathematical concepts

The systematic, rigorous study of mathematics can be said to date back around 2500 years to the Greeks. The scientific study of mathematical cognition, however, is of a far more recent date. Piaget [32] published his study on the child's conception of number in 1941 as a part of his *constructivist* theoretical framework for cognitive development. More recently, Lakoff and Núñez have emphasised the importance of metaphorical reasoning and embodiment in mathematics, effectively founding what they call the *mathematical idea analysis* - a discipline which studies both developmental and historical aspects of mathematics[1] [26]. Their central question is: how are mathematical concepts created by the human mind?

1.2 Objectives, motivation and scope

This project is a computational approach to the first of Lakoff and Núñez's central theses; namely that the first mathematical concepts are grounded in embodied, situated experiences through metaphors, and that this (innate) ability to form metaphors is what allows human beings to go beyond innate arithmetic. By "computational approach" we mean that we have a working system which *does what the*

[1] Meaning that the developmental stages, more or less accurately, repeat historical stages of mathematics, in a way reminiscent to the theory of recapitulation in biology: *ontogeny recapitulates philogeny*.

original theory claims to explain, in a way consistent with the theory itself. In our case, this means a system which develops a set of basic mathematical concepts. This type of approach requires filling in the gaps of the authors' original theory, and making assumptions about the functional mechanisms involved in the theory, in an attempt to make the theoretical practical, and the vague precise.

We will also employ concepts from other theorists, such as the notion of *action schemas* from Piaget [33]- in a way reminiscent to what Gary Drescher did in his *schema-mechanism* system [7] (see section 1.3 below), to attain the goal of the project: to show that an agent can indeed develop basic mathematical concepts from situated, embodied experiences combined with a mechanism for metaphorical reasoning.

It is important to note that this project is not a theory for how humans learn mathematical concepts. Rather, it is an application of several ideas from the field of mathematical cognition (Lakoff and Núñez's being the most central one) to artificial intelligence (AI). This means that inspiration is taken from humans, as we are the "state of the art" of mathematics learners, but less emphasis is put on the mechanisms of the system being cognitively plausible. However, interesting similarities between the system and humans will be pointed out, as they can sometimes be enlightening both for the study of humans (in cognitive science) and for the study of artificial agents (in AI).

What we want to show in this project is that an agent can develop a simple conceptual apparatus for mathematics, given a set of sensorimotoric primitives, an associative memory and a capacity for forming metaphors. We will accomplish this by having the agent operate in two different domains. These domains are based on the metaphors ARITHMETIC IS OBJECT COLLECTION and ARITHMETIC IS MOTION ALONG A PATH due to Lakoff/Núñez [24]. By making a computational approach to mathematical cognition, we also hope to contribute to the methodology of a field which is, in some respects, still in its infancy.

1.3 Related work

There are several interesting accounts of computational approaches to various aspects of embodiment and metaphor, but few, to my knowledge, that focus on mathematical cognition and the development of mathematical concepts. I mention here some related work which has, at least indirectly, influenced this project.

The Neural Theory of Language (NTL) group at Berkeley has produced work on the role of embodiment and metaphor in the development of language and concepts [9]. In particular, a triad of PhD theses were published by Regier [38], Naryanan [31] and Grady [15] in 1996-1998. The recurring theme of these projects were computational models using X-schemas - execution schemas for sensorimotoric activity - in language understanding and learning (more specifically, verb semantics). None of them were about mathematical cognition, but the embodiment approach to meaning and concepts has relevance for this project.

Gary Drescher's *schema mechanism* builds on Piaget's developmental stages of intelligence and his concept of *action schemata* [7, 33]. The result is an embodied[2] system which learns from empirical data and invents concepts, starting out with almost no *a priori* knowledge. Drescher's system reaches "some early milestones in Piagetian development of the concept of physical objects" [7, p. 11], but there is no particular emphasis on mathematical cognition and the development of mathematical concepts.

Rodney Brooks at MIT pursues what we can call a *pragmatic* approach to embodiment and AI, operating under the mantra: "The world is its own best model" [3, p. 136]. Brooks does not claim that there is any fundamental principal difference between a simulated agent and an embodied agent, but that the simplest way to make a robot develop intelligence is not to try and reproduce the world in a simulation, but rather to place the agent as a situated robot in the real world.

Other computational approaches to mathematical cognition can be found in [16, 14].

[2] Or at least *simulated embodied* - see discussion of this in section 2.3

1.4 Overview

In the following chapter, we will look at the relevant background material for the project, mention the theoretical background for conceptual metaphor, heuristic-driven theory projection (HDTP - our analogy engine of choice) and topics related to embodiment and situatedness. Chapter 3 discusses the methodology, states the hypothesis and considers ways to evaluate the results from the experiments. Chapter 4 presents the system design - that is, the system's modules and mechanisms. Chapter 5 presents a set of experiments carried out with the system and their results, followed by general discussion and evaluation of the results in chapter 6. Finally, we conclude and suggest further research in chapter 7. The appendix includes notes on the implementation of various parts of the system, such as the mechanisms of the agent, the two different domains, the interaction with the HDTP system and the experimental set-up.

Chapter 2

Background

This section describes the background for the project. We start by going through topics such as conceptual metaphor and the grounding metaphors of arithmetic. We then move on to analogy engines, where we introduce heuristic-driven theory projection (HDTP). We explain the concept of an *action schema* in our section on embodiment and situatedness. Finally, we introduce association networks as a way of formulating embodied concepts.

2.1 Conceptual metaphor

In 1980, Lakoff and Johnson introduced the theory of conceptual metaphor in *Metaphors We Live By* [24]. Lakoff and Johnson argued that metaphors are not just embellishment in language, but are also highly prevalent in everyday language, where metaphors such as HAPPY IS UP[1] and MORE IS UP form culturally coherent structures in ordinary language and thought. Some examples showing these conceptual metaphors in English are:

(2.1) She's in *high* spirits. (HAPPY IS UP)

(2.2) The number of words in my dissertation keeps going *up*. (MORE IS UP)

Furthermore, Lakoff and Johnson showed that many of our most basic conceptual metaphors have bodily experience - including sensations (warmth, coldness) and

[1] We use THIS FONT to indicate a metaphor throughout the text.

spatial orientation (up, down) - as their source domain. In this sense, our whole conceptual system is described by Lakoff and Johnson as *grounded* in bodily experience.

The basic premise of the theory of conceptual metaphor is that the metaphors we use in our language reflect the way we represent and reason about concepts. If we accept this premise, it follows that we can use linguistic evidence as more or less direct evidence for how people think. For reference, we will call this idea *the transparence hypothesis*. The transparence hypothesis gives cognitive scientists a powerful tool, as the whole realm of language usage becomes a source of data to study and analyze from a cognitive perspective.

2.1.1 The grounding metaphors of arithmetic

Lakoff later applied the framework of conceptual metaphor in the domain of mathematics, together with Raphael Núñez, in their book *Where Mathematics Comes From* [26]. Here they develop *Mathematical Idea Analysis*, a methodological framework with which they claim to be able to analyze both developmental-psychological aspects of mathematics and the history of mathematics.

Lakoff and Núñez characterise conceptual metaphor with an example from mathematics as follows[26, p. 6]:

> Conceptual metaphor is a cognitive mechanism for allowing us to reason about one kind of thing as if it were another. . . . It is a grounded, inference-preserving cross-domain mapping—a neural mechanism that allows us to use the inferential structure of one conceptual domain (say, geometry) to reason about another (say, arithmetic).

In the first chapters, we are presented with some arithmetical abilities of humans which empirical studies in developmental psychology suggest to be the *innate*. One example is the ability to instantly know how many objects are in a collection, for collections containing up to about four items - a skill known as *subitising* [26, p. 19]. From there follows an explanation of how a set of four *grounding metaphors* enable "human beings, who have an innate capacity to form metaphors, to extend arithmetic beyond the small amount that we are born with, while preserving the

basic properties of innate arithmetic" [26, p. 77]. These grounding metaphors are:

- ARITHMETIC IS OBJECT COLLECTION
- ARITHMETIC IS OBJECT CONSTRUCTION
- THE MEASURING STICK METAPHOR
- ARITHMETIC IS MOTION ALONG A PATH

In the ARITHMETIC IS OBJECT COLLECTION metaphor, the basic metaphor is that of a number being an object collection, and operations on numbers (such as addition) are the operations on object collections (such as putting them together). In the ARITHMETIC IS OBJECT CONSTRUCTION metaphor, a number is one object which consists of parts, effectively giving rise to the notion of fractions. THE MEASURING STICK METAPHOR maps a number to a physical segment - a stick - and the operations on numbers (again, such as addition) are operations on these segments (such as the concatenation of them). Finally, in the ARITHMETIC IS MOTION ALONG A PATH, moving along a path maps to operations on numbers, where a number then is the position on that path.

The individual structures of these four metaphors are not completely identical, but there is a common basis which enables the metaphors in the first place. Without going into detail on all of the four metaphors, we can see the structure of the ARITHMETIC IS OBJECT COLLECTION metaphor and the ARITHMETIC IS MOTION ALONG A PATH in figures 2.1 and 2.2. These two metaphors are the central ones in our project.

At this point one might object that we cannot explain a concept as arising from a metaphor, when the target domain of the metaphor is indeed the concept we are forming; the explanation appears circular. However, because there is an common structure between all the source domains in the four grounding metaphors which, crucially, is *independent of arithmetic itself*, the first metaphors being formed are in fact analogies[2] between the source domains themselves.

This is an important point for the order of cognitive development, even though

[2]We will discuss the distinction (if there is one at all) between analogies and metaphors in sections 3.4. For now it suffices to say that we allow ourselves to use the terms more or less interchangeably.

ARITHMETIC IS OBJECT COLLECTION

Source domain OBJECT COLLECTION		*Target domain* ARITHMETIC
Collections of objects of the same size	→	Numbers
The size of the collection	→	The size of the number
Bigger	→	Greater
Smaller	→	Less
The smallest collection	→	The unit One
Putting collections together	→	Addition
Taking a smaller collection from a larger collection	→	Subtraction

Figure 2.1: The ARITHMETIC IS OBJECT COLLECTION metaphor. Reproduced from [26, p. 55]

ARITHMETIC IS MOTION ALONG A PATH

Source domain MOTION ALONG A PATH		*Target domain* ARITHMETIC
Acts of moving along the path	→	Numbers
A point-location on the path	→	The result of an arithmetic operation
The origin, the beginning of the path	→	Zero
Point-locations on a path	→	Numbers
The unit location, a point-location distinct from the origin	→	One
Further from the origin than	→	Greater than
Closer to the origin than	→	Less than
Moving from a point-location A away from the origin, a distance that is the same as the distance from the origin to a point-location B	→	Addition of B to A
Moving toward the origin from A, a distance that is the same as the distance from the origin to B	→	Subtraction of B from A

Figure 2.2: The ARITHMETIC IS MOTION ALONG A PATH metaphor. Reproduced from [26, p. 72]

Lakoff and Núñez only briefly mention it[3]. This is the type of analogies our system will eventually form - analogies between the source domains. We will come back to this when discussing the analogy elements of the system.

2.1.2 Cross-linguistic evidence for the grounding metaphors

We find several examples of the four grounding metaphors in English[4]:

(2.3) The result is *around* 40.

(2.4) If you *put* 2 and 2 *together*, you get 4.

(2.5) 37 is *far away from* 189,712

(2.6) The chalet is close to the road, *give or take* a few hundred yards.

Examples 2.3 and 2.5 are instances of the ARITHMETIC IS MOTION ALONG A PATH metaphor, while 2.4 is an example of the ARITHMETIC IS OBJECT CONSTRUCTION metaphor. The last example (2.6) is particularly interesting, as it is an instance of either the OBJECT COLLECTION metaphor, or the MEASURING STICK metaphor, but not the MOTION ALONG A PATH metaphor[5] - which is perhaps the metaphor one would expect since the meaning of the sentence refers to spatial location (and, at least implicitly, motion). Instead the metaphor in play induces images of giving or taking away yards from some bigger collection of yards. Most importantly, though, is that it is a completely natural sentence for someone to utter.

The grounding metaphors are at least as prevalent in many other languages as they are in English. In Italian (and the other Romance languages) mathematical operations are expressed as in the following example:

(2.7) Due più due *fa* quattro.

Literally, "two plus two *makes* four", which is the metaphor of OBJECT CONSTRUCTION at play. Continuing in the same language family, the latin word *subtrahere* (from which the English "to subtract" is derived) consists of *sub-* and *-trahere* - the

[3] "... an isomorphic structure emerges across the source domains of the [four grounding metaphors]. That isomorphic structure is *independent of numbers themselves* and lends stability to arithmetic." [26, p. 96]

[4] The first three examples are from [26].

[5] Indeed, in this case it wouldn't even have to be a metaphor - it could literally be about motion along a path.

latter meaning “to draw”, suggesting the arithmetical operation of subtraction as a physical act of drawing objects from a collection.

Kinn [23, p. 70] gives the following example of the object-collection metaphor being in play in Old Norse numeral nouns:

(2.8) Þrir kattar ok tuttugu

which translates to “twenty-three cats”, or literally “three cats and twenty (cats)”. As Kinn points out, this suggests that there are two separate collections[6] of cats; one of three cats and one of twenty, and that one in fact has to perform the arithmetical operation to end up with the end result. The object-collection metaphor is still prevalent in the numerals and numeral nouns of modern Norwegian. As Kinn states: “the nominal profiles of numeral nouns are themselves evidence of the metaphor that arithmetic is object collection” [23, p.70].

Interestingly, Lakoff and Núñez do not give any examples for the measuring stick metaphor, but rather refer to classifier morphemes in Japanese noting that “Even though English does not have a single word for the idea, it is a natural human concept”. Given the transparency hypothesis, one would perhaps expect the measuring stick metaphor to be prevalent also in English if it were indeed a “natural human concept”. Following this logic, one could question the importance of the measuring stick metaphor. We shall not, however, dwell on this point.

We conclude the linguistic section with an example less related to the grounding metaphors, but one which illustrates the close connection between language and the overall cognitive apparatus of human beings. In Slavic languages, nouns succeeding numerals for 1 to 4 take on a different morphological case than nouns succeeding numerals for 5 and more[7]. For instance, in Slovak, “two oaks” is “dva *duby*” (“oak” in nominative case) , but “five oaks” is “päť *dubov*” (“oak” genitive case[8])[9]. The reason this is interesting, is that 4 is also thought to be the limit of

[6] Kinn uses the word “masses” and takes the nominal phrase as an example of object construction. We would argue that describing this as an instance of the object-collection metaphor is just as reasonable, and perhaps more intuitive.

[7] Thanks to Prof. Tore Nesset at the University of Tromsø for bringing my attention to this.

[8] This would correspond to something like “five [of] oaks” in English.

[9] The difference in grammatical categorisation can be expressed in various ways. For instance, in Russian, it is reflected in both syntax and morphology: nouns succeeding 2 to 4 take the genitive singular case, while nouns succeeding 5 and more take the genitive plural case; the numerals themselves for 2 to 4 are inflected as adjectives, while numerals for 5 and more are inflected as nouns.

what a human being can subitise[10] [26]. Categorising, grammatically, the numerals for 1 to 4 differently from the numerals from 5 and upwards might reflect this. We will not speculate too much in whether or not there exists a causal link here, but it does seem to fit well with the overall theme of cognition, language and embodiment being highly interdependent.

2.1.3 Metaphor and analogy

When discussing metaphor in the context of cognition, it is important to mention how it relates to analogy and analogical reasoning. The word "metaphor" derives from the Greek "μεταφορά" - "metaphorá", which translates to "transfer(ence)". This suggests an asymmetrical relation, and indeed in the study of metaphors one talks about a *source* domain and a *target* domain[11], where objects, relations and inferences can be mapped from the source domain onto the target domain. The study of metaphor has traditionally been a part of linguistics, literary theory and rhetoric.

"Analogy", on the other hand, is originally a mathematical term meaning "proportion" (from Greek "ἀναλογία") and was used specifically to denote similarity of ratios - such as 4:8 and 3:6 [39]. Analogies later expanded out of the domain of mathematics and the term "analogy" was re-interpreted by Kant [22, §58] as "a perfect similarity of relations between two dissimilar things". The requirement of "perfect similarity of relations" is, however, rarely fulfilled in analogies between two dissimilar domains - that is, unless we prune away all possible conflicts between relations in the two domains and re-define the domains with only the perfectly similar relations left, in which case the perfect similarity is trivial. This is a part of the problem Hofstadter [18] calls the key to analogy-making: "extracting the gist" of the domains (for further discussion on this, see section 6.2).

Gentner [11], who developed the highly influential *structure-mapping theory (SMT)* of analogy, describes analogies as comparisons between two domains where relations are mapped, but where only few (or no) predicates (i.e. one-place rela-

[10] Dehaene [6, p. 68] states that "The numbers 1, 2 and 3 seems to be recognized without any appearance of counting", thus excluding 4 from these first subitisable numbers. However, this is based on experiments involving collections of different cardinalities, with reaction times getting longer and error percentage getting increasingly higher for collections with higher cardinalities. Thus, it may not be a matter of be-or-not-be for the role of 4 as a subitisable number.

[11] The terms "source" and "target" could themselves be thought of as metaphors.

tions) are mapped. Metaphors are described as being like analogies[12], except that more predicates are usually mapped, in addition to relations. The mapping rules of metaphors are also described by Gentner as being less regular. One could suspect that this comes from the metaphor's rhetorical role in art and literature. Figure 2.3 shows a diagram illustrating Gentner's descriptions of metaphor and analogy.

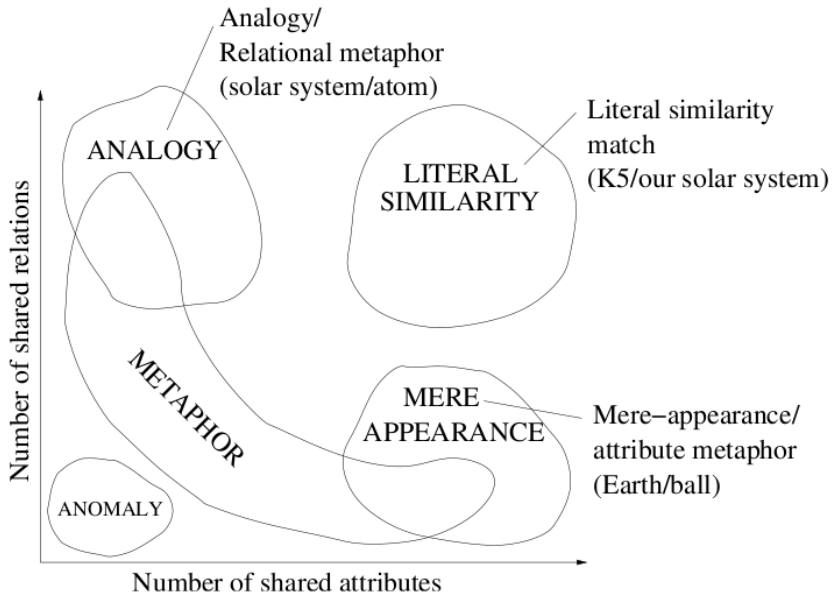

Figure 2.3: Gentner's proposed illustration of the difference between metaphor and analogy. From [13], through [39]. While the source and target domains in metaphors often share attributes, analogies are based purely on the relational similarity of the domains - giving rise to a possible synonym for analogy: *relational metaphor*.

It is also common to talk about a *source* domain and a *target* domain in the literature on analogy, just as with metaphors. The relationship is not symmetrical in that for instance transfer of relations or objects usually go from the source domain to the target domain, but now the other way around.

One could argue that the notion of conceptual metaphor is equivalent to analogy, since it transposes metaphor from language to the overall conceptual system.

[12]This is consequently a metaphor in itself - or perhaps a *simile* cf. [29]

In fact, it has been claimed that a large number of metaphors can be reduced to analogical reasoning [17]. An example given is the metaphor:

(2.9) Gills are the lungs of fish.

Which can be reduced to the underlying analogy:

(2.10) Gills are to fish as lungs are to mammals.

Following this line of argumentation, we will, for instance, also use the term "analogy" as well as "metaphor" to refer to the common structure between the source domaims of the grounding metaphors (see section 3.2 for a clarification of terminology used). Furthermore, we will treat the capacity for making metaphors, which Lakoff and Núñez describe as innate in human beings, as a part of a more general cognitive apparatus for analogy reasoning. In such an apparatus, there are also mechanisms for generalisation and association.

Of course, saying that conceptual metaphor is equivalent to analogy does not make the overall *theory* of conceptual metaphors superfluous. The hypothesis that one can study metaphors in language to reveal underlying conceptual metaphors is still a prerequisite for the approach to mathematical cognition that we take here.

2.1.4 Analogies in mathematics

Since the concept of analogy is originally a concept taken from mathematics, is is ironic that the literature on analogies in mathematics historically has been so sparse [39]. George Polya was for a long time the only established mathematician to systematically study mathematical reasoning[13], and one of the topics he wrote about was analogies, which he regarded as central for mathematical reasoning [36]. Apart from Polya, metaphors and analogies in mathematics have so far mainly been studied by educators (see e.g. [30, 43, 44, 34, 2]).

Another striking thing about analogies in mathematics is that most classic theoretical approaches to analogies have not been very successful in mathematics. Schlimm [39] gives several arguments for why this is the case, the central one being that mathematics is an *object-rich* domain, where there are more objects than

[13] Or who did so also commenting on the cognitive aspects of mathematics.

relations in the domain, as opposed to a *relation-rich* domain, where there are more relations than objects. Since most of the domains conventionally studied in analogical research are relation-rich, Schlimm argues that the classical theories are not well-enough equipped to tackle object-rich domains. He then goes on to formulate an axiomatic approach to analogies in mathematics, after noting that "for the kinds of domains that are frequently investigated in mathematics the formulation of analogies in terms of axioms is more adequate than the use of explicit mappings" [39, p.161].

We think that Schlimm's axiomatic approach to analogies in mathematics is appropriate for higher levels of mathematical reasoning, but we shall see that the analogies based mainly on the internal structure of relations work well for the development of the very first mathematical concepts, where there typically are few objects (or states) involved.

2.1.5 The usefulness of analogical reasoning

Why do we make analogies? First of all, analogies can work as an inference tool. We could have a mapping of the type:

$$s_1 \to t_1$$

$$s_2 \to t_2$$

$$\ldots$$

$$s_n \to t_n$$

where s_i and t_i represent facts about the source domain and target domain, respectively, and the arrows represent mappings (rather than logical connectives). If we then somehow discover:

$$s_{n+1}$$

then the analogical inference is to conclude:

$$t_{n+1}$$

This is not a logically sound inference, and consequently it sometimes introduces new information (just like an inductive inference).

Second, we can introduce new objects, predicates and relations into the target domain, based on the ones we find in the source domain (which do not already have a counterpart in the target domain). An example of this occurs in the LOVE IS A PARTNERSHIP metaphor due to [26, p. 46]. Here we have mappings such as:

LOVE RELATIONSHIP IS BUSINESS PARTNERSHIP

Source domain		*Target domain*
BUSINESS		LOVE
Partners	$\rightarrow$	Lovers
Partnership	$\rightarrow$	Love relationship
Wealth	$\rightarrow$	Well-being
Smaller	$\rightarrow$	Less

These are the basic mappings from which new concepts can be introduced into the target domain. These new concepts extend the metaphor with the following mappings:

Work put into partnership	$\rightarrow$	"Work" put into relationship
Profit from the partnership	$\rightarrow$	"Profit" from the relationship

We will see an example of a extensions to a target domain in mathematics resulting from our experiment in section 5.4.

2.2 Analogy engines

From the theoretical work on analogies in cognitive science came implementations which automatically formed analogies between two domains, following the theoretical principles of the model on which they were built. The best-known analogy engine is probably the Structure Mapping Engine (SME) [8], based on Gentner's Structure Mapping Theory (SMT) [11]. SMT is psychologically motivated, meaning that it is based on human analogy making.

Another notable analogy engine is Copycat, developed by Mitchell and Hofstadter [20]. One of the goals of the Copycat project was to make the process of forming the analogy itself psychologically realistic. To include as many aspects of analogy formation as possible, the input was restricted to a very simple "micro-domain" of strings, where the problem was to guess the next letter in a sequence.

Although we do not give ourselves the restriction of developing a model which is psychologically realistic, we do agree with Mitchell and Hofstadter's approach: it is better to understand the whole process of analogy formation in a simple domain, than it is to understand only small parts of the overall process in more complex domains.

2.2.1 Heuristic-driven theory projection

Heuristic-driven theory projection (HDTP) is a formal logic-based symbolic approach to analogy making [40], as opposed to SMT, which is a first and foremost psychologically motivated model and a graph-based approach [41]. The main mechanism by which HDTP works is generalisation by anti-unification. The basic rules for anti-unification in HDTP are:

- Two formulae $p(a)$ and $p(b)$, are anti-unified by $p(X)$[14] (first-order unification), and
- Two formulae $p_1(a)$ and $p_2(a)$ are anti-unified by $P(a)$ (second-order unification).

Readers familiar with unification in logic programming or type theory will see that these rules are just the inverse of the unification rules, which is where anti-unification gets its name from.

When two domains are specified by first-order logic formulae, giving a theory Th_1 and Th_2 for each of the two domains, HDTP tries to anti-unify the formulae of these theories in order to reach a more general theory Th which is consistent with the formulae of the original theories. If this can be done, HDTP outputs the analogy resulting from the generalisation. If the best-fitting general theory predicts

[14] Here we are using the syntax of Prolog where lower case letters (such as *a*,*b*) represent constants, while capital letters (such as *X*) represent variables.

the transfer of a term from the source to the target domain, this transfer is a part of the output given. This is where the asymmetry between source and target domain is evident.

One of the things that make HDTP interesting is that it has more flexibility than other analogy engines. In addition to the very basic rules given above, there are rules for such things as argument permutation and argument insertion, and associated *costs* with these rules. If a rule is being used to form an analogy, then the cost of that rule is added to an overall cost of the analogy (see [42, 40] for more details). This is a nice feature, as it enables us to put different restrictions on which analogies can be made, as well as potentially empirically test how well the cost scheme of HDTP corresponds to a possibly existing cost scheme in human subjects (see further work in section 7.2). Also, the fact that an analogy is described as a more general underlying theory of the source and target domains, gives us a nice parallel to how we imagine mathematics as being formed as a concept through "abstraction by analogy".

2.3 Embodiment and situatedness

When discussing embodied cognition, it is common to draw a distinction in the following simple way: an agent which has sensorimotoric interaction with the real world (i.e. the world that we ourselves interact with) is called *embodied*, while an agent which only interacts with the real world through, say, a keyboard and a screen is called *disembodied* [45]. However, we could still talk of an agent existing in a closed-loop simulated world, in which case the situation is less clear-cut. The interaction between agent and environment in such a simulated world will not be as rich as real-world, but the system as a whole can be said to inhibit some properties of embodiment. Specifically, and important for our purposes, representations can still be built on sensorimotoric primitives, such as ways of *acting* on the environment and *observing effects* of those actions.

We will refer to an agent in such a simulated world as being *simulated embodied*, or simply, *simbodied*. Some would argue that any talk of simulated embodiment is missing the point of embodiment altogether, and that such an agent is simply

disembodied. Without getting too deeply into philosophical waters, we stress that by employing the notion of simbodiment, we are following what we could call a *pragmatic* view of embodied cognition: having a body helps an agent to build representations from sensori-motoric primitives, as well as enabling only certain types of perception of, and action on, the environment, thus restricting the possible input and output an agent can have.

2.3.1 Action schemata

One of the ways the system will incorporate embodiment is through the use of *action schemata*, a term taken from Piaget [33]. Piaget explains his concept of an action schema as follows: "We shall apply the term 'action schemata' to whatever, in an action, can thus be transposed, generalized, or differentiated from one situation to another: in other words, whatever there is in common between various repetitions or superpositions of the same action" [33, p. 7].

The way we will use the term "action schema" is slightly different from that of Piaget. Our approach can be said to go the other way around: the agent will first classify similar action sequences together under their contexts and outcomes. It will then cluster the set of these pairs of context and outcome together based on a measure of similarity (see chapter 3 for details). This resulting cluster is what defines the action schema.

We emphasize the role of action schemata in structuring the world, forming concepts and associating those concepts with each other. Furthermore, since the agent is acting in deterministic environments, an action schema simply contains all action sequences which have the same outcome, given the same context (again, consult chapter 4 for more details on this).

2.4 Association networks

In order to address what it means to maintain a concept of something, we will refer to *association networks*. These are networks where the nodes are things like input from various modalities, motoric action, or other subnetworks. An example

of the embodied CONTAINER concept is shown in figure 2.4. The idea of using associations as a basis for concepts is based on the intuitive (and recursive) notion that knowing a concept C involves knowing which other concepts C is associated with, and how they are associated. Lamb [27] has used these types of networks to explain how concepts are represented in the brain. Using network theory in the field of mathematical cognition is, however, a more recent idea explored by Mowat and Davis [28].

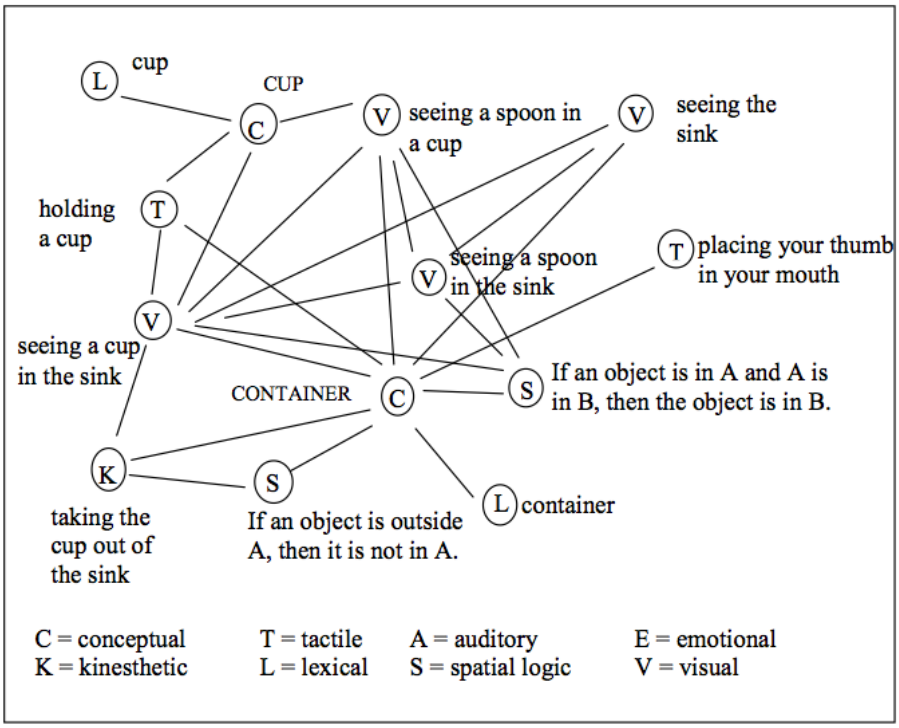

Figure 2.4: A graph representation of the concept CONTAINER from [28, p. 11].

The network in figure 2.4 shows how embodied concepts are cross-modal. It is not necessarily well-structured; the point is that the activation of some concepts triggers the activation of others. Furthermore, the nodes in it differ in their simplicity of being represented in computational terms.

2.5 Summary

We have seen how conceptual grounding metaphors are taken to be the basis for mathematical concepts. Furthermore, there are analogy engines available which do at least a part of the overall analogy-formation process; the one we choose for our project being heuristic-driven theory projection (HDTP). Embodiment and situatedness are topics which relate to the "grounding" of concepts, with the notion of an *action schema* being central for our purposes, as well as the use of association networks, which can describe rich embodied concepts.

Chapter 3

Methodology

In this chapter, we discuss the central hypothesis of the project, the terminology we employ, how we can try to test the hypothesis in a reasonable way, and how we evaluate the results of the experiments we have run.

3.1 Hypothesis

The background in chapter 2 leads us to the central hypothesis of the project:

> *An embodied agent can develop basic mathematical concepts given a set of sensorimotoric primitives in more than one domain, mechanisms for association, and a capacity for forming metaphors.*

This is an "existential" hypothesis; it predicates that there is at least one agent-mechanism scheme such that the agent develops basic mathematical concepts. This means that we cannot falsify it empirically, which might make it sound unscientific as it appears to violate the widely-accepted falsification as criterion of demarcation [37]. However, we could easily have stated the negation of the hypothesis, and then worked to try and falsify it. This might seem trivial, but it is important to emphasise, since it makes explicit what we can and cannot conclude if we in this project would not able to make a system which developed any basic mathematical concepts. We would then not have showed anything universal from a purely ontological perspective, one way or the other. Crucially, we would *not* have shown

that an embodied agent *cannot* develop mathematical concepts given sensorimotoric primitives, associative memory and a metaphorising capacity.

In the case of success, we would have an overall framework for developing basic mathematical concepts in agent, and it also would suggest the existence of other mechanisms for other domains, which in turn could lead to a more general framework for computational approaches to the developmental aspects of mathematical cognition.

Another important thing to point out, is that the hypothesis does not mention anything about humans, nor does it mention anything about cognitive plausibility. Even though the theorists (specifically [26]), whose work this project in large part builds on, make claims about humans and the cognitive plausibility of the mechanisms, that does not mean that we are obliged to do so. Indeed, these theorists might very well be wrong in that humans develop mathematical concepts through the formation of metaphors, and our hypothesis can still be true. This is important to point out, and it shows that this project belongs primarily to the field of artificial intelligence, and only secondarily to the field of cognitive science.

3.2 Terminology

In stating the hypothesis above, a short clarification of how we use the terms "embodied", "concept", "associative", "sensorimotoric primitive" and "metaphor" is in order. This section makes explicit what we intend when using these terms in the text. Readers interested in a more detailed discussion on the background of some of these terms are referred to chapter 2.

3.2.1 Concepts and associations

We will not use a rigid definition for the word "concept", but rather refer to compound structures of *associations* as concepts, as described in section 2.4. By "associations" here, we mean both embodied concepts in the form of *association networks*, but also *abstractions* (similar situations which can be abstracted to an overall schema of situations are associated with each other through that abstrac-

tion) and *analogies* (domains which can be described as analogical are associated with each other through the analogy).

Consequently, to say that our agent "develops a concept" means that it develops a structure of associations; specifically a structure consisting of association networks, abstractions and analogies. This makes it possible for us to inspect and evaluate what we will call a "concept" that the agent has developed.

3.2.2 Metaphor and analogy

Following Gentner's [13] description of the difference between metaphor and analogy, the similarity which exists between the source domains in Lakoff and Núñez's four grounding metaphors is an analogical one, and not a metaphorical similarity. This is because it is first and foremost a structural similarity, and not one based on shared attributes or appearance (see figure 2.3). However, since Lakoff and Núñez refer to all of their mappings as metaphors, we shall also call employ the word "metaphor" when talking about analogies. Their notion of a "metaphorising capacity" can also be seen as encompassing both metaphors and analogies, although others (such as [12]) would refer to this capacity as a capacity for analogical reasoning.

In brief, we will not draw any strict line between metaphor and analogy, and refer to the output of HDTP both as an analogy and as a metaphor, depending on the context.

3.2.3 Embodiment and sensorimotoric primitives

By an "embodied" agent, we really mean a *simbodied* agent, as described in 2.3. This means that the agent is acting in a simulating world, and that the agent is itself simulated. The sequences of actions it performs are composed of *sensorimotoric primitives*, such as 'take away a block' and 'add a block' in the Object Collection domain, and 'move one step forward' and 'stay in the same place' in the Motion Along A Path domain.

There are four properties of these sensorimotoric primitives worth mentioning. Firstly, we take these primitives to be the most basic movements the agent can do -

hence their name. Secondly, we take them to be given; i.e. some preliminary stage might have taken place where the agent learned these primitives, or they might be hand-coded by a programmer - as is the case in this project. Thirdly, the agent can separate and identify the primitives, meaning it never confuses 'take away a block' with 'add a block'. Lastly, the outcome of the sensorimotoric is deterministic, given a deterministic world; in other words, the agent never fails in executing a primitive, or to borrow two Chomskyan terms from linguistics [4]: its competence is equal to its performance.

It is important to point out that many of the parts of the system which are simulated, could be implemented in a real-world physical robot. Of course, a real world introduces more challenges. We have abstracted away these challenges, but this is mainly to keep focus on what is important for our purposes, namely the development of mathematical concepts. For instance, in our system, the agent checks which state it is in by inspecting a simple string given as input. In a real-world, the agent would need mechanisms for object recognition, classification, etc., if we wanted it to intereact with a real-world version of our simulated worlds.

Noting this, we still allow ourselves to draw conclusions which go further than the purely simulated world. There are already fairly good mechanisms for object recognition, etc., so we could imagine having this part of the system outsourced to some third-party. There is always the risk that one is abstracting away too much, and losing critical parts of information which could be useful in forming representations, but this is something we have to accept in a project with such a short time-frame.

3.3 The two domains and the environments representing them

We have focused on two of the original four grounding metaphors by Lakoff and Núñez: the ARITHMETIC AS MOTION ALONG A PATH metaphor and the ARITHMETIC AS OBJECT COLLECTION metaphor. These were chosen because their source domains are (at least superficially) the most divergent of the source domains in the four grounding metaphors - meaning that we could implement two toy worlds that were sufficiently different for it to be interesting from an analogical point of view.

How to implement these two domains as environments for the robot to interact with is an important part of the methodology. Clearly, the environments must be made in such a way that there is an underlying similar structure for the agent to discover and then base its analogy on, but they cannot be so similar they formally are just the same model, or program, expressed with different variable names, etc.

Instead of worrying too much about what the "essence" of the object-collection domain or the motion-along-a-path domain is, we decided to implement two different environments with different states and different possible action sequences, in addition to a slightly different mechanisms for how to form action schemata. We also set up the intra-domain experiments in different environments so that the agent could transfer knowledge built up in one environment to the other, and vice versa.

3.4 Stages of analogy formation

An important part of the methodology is to establish which stages are involved in the process of forming an analogy, and how we tend to resolve the problems in these various stages. Following [39, p. 150], we can divide the process of drawing analogies into seven stages:

1. Representation: How is the information about the analogies coded?
2. Retrieval: How and when is the source analog retrieved from memory (and, perhaps, with which purpose)?
3. Elaboration: Which parts of the source analog are retrieved, and how are these parts chosen?
4. Mapping: How is the mapping between source and target domain established?
5. Inferences: Given the mapping, which inferences about source and target domain can be made, and how?
6. Evaluation: How are the inferences evaluated?
7. Learning: How are the new problem solutions, given from the inferences made, learned?

Like most analogy engines, HDTP works first and foremost on the mapping stage, but with its calculation of costs of analogies, it also has a way of evaluating its own analogies. Furthermore, both inferences can be said to be done by it since it transfers objects, predicates, relations and statements from the source domain to the target domain, if appropriate. This means that to make a system which does the whole process of forming - and using - analogies, we need to develop mechanisms for representation, retrieval, elaboration and learning outside of HDTP. Especially the representation and elaboration stages pose an interesting challenge.

In our case we will use mechanisms which are somewhat tailor-made to the discrete worlds our agent will operate in, but which also have generalisable characteristics.

3.5 Evaluation

The natural way to test whether or not the agent succeeds in developing concepts is to inspect the structures of associations it has formed. For the abstractions it makes, this means checking which situations it clusters together (effectively forming an abstraction), and seeing if these clusters correspond to anything one could describe as a mathematical abstraction in a particular domain. We do the same with the association network; that is, we inspect the associative connections between nodes, and see if the overall network can be described as some reasonable concept.

As for the analogies, we evaluate success using three heuristic measures:

1. Does the agent form the analogies we expect it to form (these can be specified in advance)?
2. Does the agent form *only* these analogies?
3. If it does form other analogies, are these analogies meaningful[1] or non-sensical?

Criteria 2 and 3 are measures which have been largely ignored by research in analogies, and relates to what Hofstadter calls “extracting the gist” of a concept [18]. If

[1] By “meaningful” we mean “corresponding to some reasonable (mathematical) concept we are familiar with”.

we feed an analogy-engine with two hand-made sets of statements coming from two different domains where we expect it to discover analogies, then it will often find the expected analogy, simply because we have given it the important parts of the domain by hand; in Hofstadter's words, we have "extracted the gist" for it. Thus, a complete analogy-system which takes all seven stages of analogy-making mentioned in section 3.4 into account, needs to be given not just the parts of the domain important for the analogy, but also the "insignificant" parts. This brings with it a risk of forming non-sensical analogies, but is much more reasonable way of evaluating the success of a complete[2] analogy-making system.

3.6 Summary

In summary, we have seen that our hypothesis is not of a typical character. It required a clarification of terms such as "concept", "metaphor" and "embodiment", before we could move along to discuss other aspects of the projects. These other parts included how to represent the environments, which other mechanisms are needed in analogy formation and how these relate to HDTP, and finally, on what grounds we could evaluate the outcome of the project.

[2] "Complete" here means a system which performs all stages of analogy-making described in section 3.4 - *not* necessarily one which can make any type of analogy.

Chapter 4

System design

In this chapter, we start off by giving an outline of the system and the two environments for the agent to interact with. The mechanisms of the agent, such as forming association and metaphor formation, follow a fairly general scheme - making them more or less independent of the environments. Therefore, we can describe these mechanisms in a general way for both domains, but by using examples from the environments for illustrative purposes.

After describing the environments, we make explicit the innate capacities of the agent, describe how its first concepts are formed from sensorimotoric primitives, describe the simple mechanisms for active exploration of the domains, outline the associative mechanisms involved in building action schemata and forming association networks, and finally describe how it forms metaphors and transfers knowledge from one domain to the other. For details on how the various parts of the system we refer the reader to the appendix.

4.1 Outline of system

The system as a whole consists of an agent being placed in two simulated worlds: the Walker world and the Bin world. The agent has some simple actions it can perform in each of the worlds, it has an associative-memory module - consisting of one mechanism for clustering states together and one for Hebbian learning of the co-occurrence of events - making it able to connect states of the world together

through its associations, and it has a capacity for forming metaphors (which is done by passing on the output from the associative-memory module to HDTP).

4.2 The environments

The environments are based on the ARITHMETIC IS MOTION ALONG A PATH and ARITHMETIC IS OBJECT COLLECTION metaphors, and are called *the Walker world* and *the Bin world*. One of the challenges of designing these worlds is to make them similar enough so that we can use the same general framework for both of them, and at the same time different enough for the discovery of analogy to be interesting. Below follows a description of the two worlds with some details on what they look like.

4.2.1 The Walker world: Arithmetic as motion along a path

The Walker world is a very simple world with the agent being able to be in one out of four states. These states are recognised by landmarks - an *apple*, a *ball*, a *coin* and a *dinosaur*. We will denote the states by their associated landmarks, or simply by their first letter (a, b, c or d) - meaning that for instance *apple* (or a) refers to the state where an apple is observed.

The agent can perform three distinct actions: *stay*, *fwd* or *bwd*. We can think of the landmarks as being arranged on a line with *apple* being the starting state, and when the agent performs *fwd* it ends up in b. From b the *fwd* action takes it to c, and from c it takes the agent to d. The *bwd* action is the inverse, so from d the agent ends up in c by performing it, and so on. The *stay* action just leaves it in the same state.

The agent performs these actions in sequences of up to length 4. For instance, *fwd-stay-fwd* is one such action sequence. We use the maximum number of 4 because this is the limit of subitising. There is no evidence, to our knowledge, one way or the other, from research in developmental psychology which would suggest an analogous notion to subitising in planning or motor execution, but since we should pick an upper limit, the subitising range becomes a natural choice.

After the completion of a full action sequence, the agent observes which state it has ended up in. It then records in its perfect, non-volatile, memory[1] the transition as a triple (s_1, s_2, A), where s_1 is the state it started in (the context), s_2 is the state it ended up in (the outcome) after executing the action sequence A.

4.2.2 The Bin world: Arithmetic as object collection

The Bin world is a world with blocks put into different bins. These bins can be seen as collections, where the boundary of a collection is given naturally by the physical borders of the bin.

In our world, the agent focuses on one bin only and can perform two distinct actions: *remove* and *add*[2] - as their names would suggest, *remove* has the effect of removing a block from the bin, while *add* has the effect of adding a block to the bin. Removing a block means placing it in another bin, and adding a block means taking a block from another bin and putting it in the bin the agent focuses on.

The agent has the ability to recognise, and separate between, first and foremost five states of the world; when the bin has no blocks, and when it has one, two, three or four blocks. These states are given the names "none", "one", "two", "three" and "four". In addition to these five, there is another type of recognisable state - "many" - which is not actually one state, but the set of all states where there are more than four blocks in the bin. We shall therefore refer to "many" as a *pseudo-state* [3]. Again, the limit of four blocks is taked from the literature on subitising (e.g. [26, 6]), albeit this time in the domain where notion of subitising originally comes from.

4.3 Innate capacities of the agent

The agent is given the following by us as designers:

- *Sensorimotoric primitives*. This corresponds to the single actions it performs in sequences known as action sequences, and are the basis for any concept

[1] This, of course, represents a clear divergence between the agent and human beings.

[2] *Add* is of course already a mathematical operation, but here we take it to mean the process of putting objects together.

[3] The agent itself treats "many" as a "don't-care" state, reflecting its lack of interest for something it does not really recognise as a single state. Here we can draw a parallel to findings in developmental psychology showing that fixation times reduces when children are presented with inconsistent behaviour for collections above the subitisable range (see e.g. [26, p. 15-16]).

formation. These are *fwd*, *bwd* and *stay* in the Walker world, and *add* and *rem* in the Bin world.

- An ability to *recognise* and *separate* certain states from each other. In the Walker world, this corresponds to recognising, for instance, the apple as a landmark and noting that it is different from the ball, coin and dinosaur toy; in the Bin world this ability corresponds to *subitising* - the ability to separate between small collections of different cardinalities - but it also includes recognising a collection of three blocks as the "same" collection as another collection of three blocks, even though the blocks might be arranged differently, etc.

- An *associative memory* with which it perfectly records and recalls events in the form of triples containing a context, an action sequence and an outcome. This memory is *associative* in three respects: i) the agent can associate action sequences with each other based on equal contexts and outcomes (e.g. the sequence *fwd-fwd-bwd* is associated with *fwd-stay* because, given the same context, they also have the same outcome); ii) the agent can associate *context-outcome* pairs with each other based on similarity of actions used to get to the *outcome* given the *context*; and iii) the agent has a mechanism for *Hebbian learning* in which it associates events based on their co-occurrence[4].

- A *metaphorising* capacity. This refers mainly to the mapping, inference and evaluation stages of the stage list in 3.4, but together with the sensorimotoric primitives and the associative memory, the agent has a metaphorising capacity which can be said to incorporate the other stages: representation, retrieval, elaboration and learning.

These are the agent's *a priori* knowledge and abilities. In chapter 5 we will show that the agent develops new knowledge from these mechanisms.

[4]We shall see later (section 4.4.2) that it is not really co-occurrence, but rather the expectation of one event succeeding another, that the agent builds associations on.

4.4 Preliminary associations

The agent has two main mechanisms for association which will be called *preliminary associations* because they are assocations which can be formed before any metaphors have been formed. The first is a mechanism which enables the agent to cluster together *context-outcome* pairs based on the similarity of the action sequence performed to get from the context to the outcome. This mechanism is crucial in that it gives the agent its first abstractions which in turn form the basis of any metaphorical reasoning being done.

The second mechanism is a mechanism which relates events[5] to each other through subsequent co-occurrence, and which gives the agent a sense of locality of the states: which states are "close" to each other - i.e. which states tend to succeed (or precede) other states.

4.4.1 Action schemata

In both the Walker world and in the Bin world, the agent bootstraps its first concept through its sensorimotoric primitives and the execution of action sequences. This is done by viewing a set of equivalent action sequences as an *action schema*. An action schema in this context is thus simply the set of all action sequences which yield the same outcome s_2, given the same context s_1, for all contexts. The agent associates with an action schema all the *context-outcome* pairs that the schema is defined on. Keeping in mind that the two worlds we are dealing with are both deterministic worlds, this gives the agent a robust way of structuring its environment.

For instance, in the Walker world, the action schema which takes the agent from state *apple* to state *ball* includes (but is not limited to) the sequences *movefwd*, *stay-movefwd* and *movefwd-movefwd-movebwd*. This action schema is also associated with moving from *ball* to *coin*, and from *coin* to *dinosaur*; we can think of it as all the action sequences which takes the agent exactly one step forward. The agent does not have any action schemata stored *a priori*. It has to discover them by exploring its environment and record the outcomes of its actions.

[5] These events correspond to the observable states in our implementation, but could in theory include other aspects, such as the agent's own actions.

4.4.2 Hebbian association network

The agent also associates events using the *Hebb rule*, a simple rule used in computational neuroscience to describe the change in synaptic weights between neurons [5]. We do not operate at the level of neurons in our agent, but rather let the agent use the Hebb rule at a higher level: instead of changing the synaptic weight between neurons, the agent changes the association strength between events according to the Hebb rule.

There are many variants of the Hebb rule, but the one we use is described by the following differential equation:

$$\frac{dW}{dt} = \eta \mathbf{x}\mathbf{x}^T \tag{4.1}$$

where W is the weight matrix containing entries w_{ij} denoting association strength between events i and j, η is a learning-rate parameter and $\mathbf{x}$ is the input pattern. The input pattern consists of -1's and 1s, indicating non-active event or active event, respectively. An example input pattern is of the type $(-1, 1, 1, -1, -1, -1)^T$, which represents events 2 and 3 as active, while events 1,4,5 and 6 as non-active. Initialising all weights to 0, we can then use equation 4.1 to update the weights over time as new events are observed. From the formula we can see that events which are active (and non-active) at the same time will build up associational strength.

The actual "events" that the agent uses this association mechanism for is the observation of states. Section 5.1.2 describes how this was done in the Bin world.

4.5 Situated, active exploration

What we call "situated, active exploration" refers to how the agent explores the state space of the environments. It is situated because the agent is placed in the world, being able to act in it and observe the effects. It is active because there is a mechanism by which the agent explores the space.

4.5.1 The will to move - deciding what to do

The action sequences the agent performs in the two domain are formed using two rules which together gives us a generative probability distribution of action sequences. First of all, the probability of picking an particular action a is uniform, so for N different actions:

$$P(Action = a) = \frac{1}{N}$$

Second, the agent is biased towards performing shorter action sequences. We express the probability of the length of the action sequence being n using a recursive rule:

$$P(Length = n) = \begin{cases} n = 1 & \frac{1}{2} \\ 1 < n < M & \frac{1}{2}P(Length = n - 1) \\ n = M & P(Length = n - 1) \end{cases}$$

where M is some fixed maximum length of action sequences (in our system $M = 4$).

These are the two simple rules which govern the agent's *will to move* in the worlds. However, in both the Walker world and the Bin world, there are situations which require an *ad hoc* solution to the problem of which action sequences are doable - namely, the end-cases. In the Walker world, this is when the agent reaches the *apple* landmark, and wants to move backward, or when it reaches the *dinosaur* landmark and wants to move forward. In the Bin world it occurs when the agent tries to pick up a block and there are no blocks left in the bin.

In these end-cases, the execution of the action sequence simply halts when the problem occurs. Then the robot records *only the sequence up to the problem occured*. Say the agent is in state *coin* in the Walker world and is about to execute *fwd-fwd-bwd*. In this case, the agent will execute the single action *fwd* and the halt, as it is in state *dinosaur* and cannot execute another *fwd* action. This puts a bias on which action sequences are actually executed, with respect to our original distribution of action sequences. The resulting distribution of transitions between states *with bias*

is shown in figure 4.1 for the Walker world.

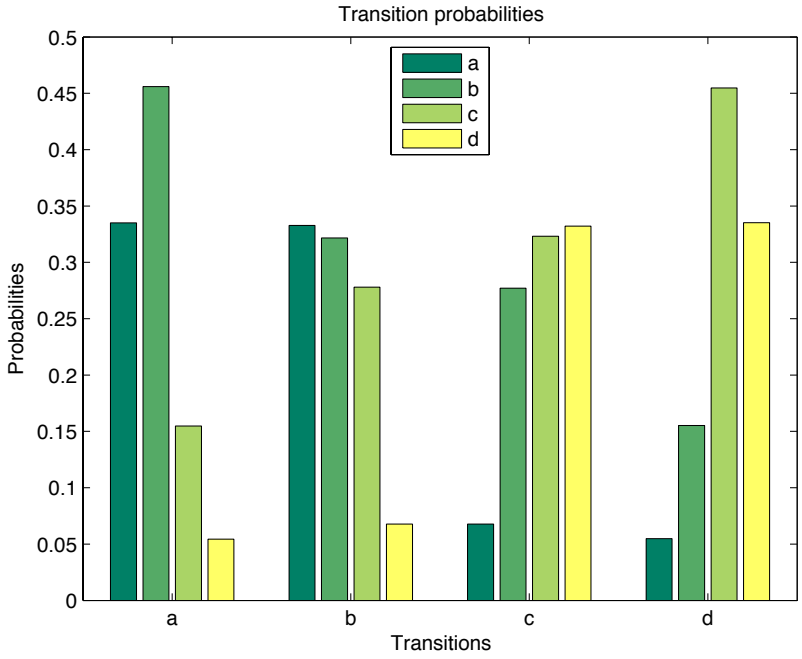

Figure 4.1: Diagram showing the probabilities of the different transitions in the Walker World. The x-axis shows which state the agent is moving from, and the colour shows which state it is moving to. For instance, the yellow bar farthest to the left shows the probability of transitioning from state *a* to state *d*. The probabilities were obtained empirically by letting the agent move around executing 4000 action sequences picked from the distribution described in the text, for 200 different seeds. Of course, the probabilities could also be computed analytically.

4.5.2 Memory hints

Sometimes the agent breaks out of its pattern of random-sequence execution; this occurs when it gets a *memory hint*. A memory hint is when the agent performs an action sequence it has performed before in a different context and recalls that situation. We can relate this to co-activation of neural structures in biological organisms as follows. Observing the new state transition T_1 activates a certain neural structure N_1 (as, say, a visual memory), which is accompanied by the activation of

another neural structure A for the action schema (this is of a sensorimotoric character). A is already associated through Hebbian learning with a neural structure N_2 (for the previous transition T_0), and so activating A activates N_2.

Of course, this is a highly simplified account of how association works in neural systems, but the principle remains the same. A first step to making the recall mechanism more biologically plausible is to parametrize it with a probability of recall, making memory hints imperfect. We have chosen to keep recall perfect for our agent: if it has performed the given action sequence before in a different context, it will recall the situation. This design choice was made because we have no overall restrictions of biological plausibility, and simply because it speeds up the learning.

The role of the memory hint in the system is to make the agent "feel" like there is a similarity between the transition (*a*,*b*) and (*c*,*d*), where (*c*,*d*) is the recalled transition. This similarity will then be put to a test by the agent. In the Walker world, this means going back to the state it just came from (in our example this is state *a*), and then pick another action sequence A_2 from the set of sequences stored for (c, d) *which is not stored for* (a, b). Execute A_2 from state *a*; if the agent ends up in state *b* also with this action sequence, then it clusters together (*a*,*b*) and (*c*,*d*) in an action schema; if it ends up in another state, then it leaves the transition pairs unclustered. Of course, since this world is deterministic, and equivalent moves are always equivalent, for all transition pairs, it will always cluster together the pairs if it ends up in the memory-hint situation. In other words, its memory-hint hypothesis is always verified. In a non-deterministic world, the agent could employ a more sophisticated mechanism for testing these memory hints, by for example incorporating the notion of *degree of belief* from Bayesian probability theory [21].

4.6 Metaphors

Metaphors are formed by the system using heuristic-driven theory projection (HDTP) - a mechanism for forming metaphors based on first-order logic representations of domains. These first-order logic representations are made in the first stage of forming action schemata, described above (for more details on this process, see sections 5.1 and A.1). Then, these representations are given as input to

HDTP.

After the preliminary stage of building up action schemata, the agent has a set of predicates[6] representing the schemata. These predicates are defined on all the context-outcome pairs on which the action schema is defined. Essentially two different different types of metaphors are formed using the predicates representing action schemata. The most obvious is the metaphor between the two different environments which represents the analogy between the two different source domains of Lakoff and Núñez's grounding metaphors.

4.6.1 Connecting the two domains by metaphor

The metaphor connecting the two domains is formed using the predicates from each of the two environments. We take the two worlds to be superficially distinct enough for the agent to recognise them as being two different domains. This is reasonable, since what it observes in the Walker world are the four landmarks, while it in the Bin world observes only collections. It is never the case that, say, a ball results from adding a block to a collection. Nor does the agent perform the same type of sensorimotoric primitives in the two domains.

Thus, we take it for granted that the agent never confuses the Bin world with the Walker world.

4.6.2 Intra-domain metaphors

There is, however, also the possibility of forming metaphors within a single environment. Since the world we as humans live in does not always come readily segmented into fixed domains, it is reasonable to assume that there are underlying mechanisms for such segmentation available in our cognitive apparatus. Similarly, the agent should also have at least one such mechanism implemented.

The challenge here is then to split one single environment into two domains. More specifically, we need to segment the predicates which result from the abstraction mechanism in a single environment into two partitions. We tried two different

[6] From now on, we'll use the term "predicate" to refer to relations as well. This means that a predicate can have more than one argument.

mechanisms for segmentation: *predicate-cluster isolation* and *non-singular segmentation*. The results of these segmentations, and the main resulting metaphor - the BACKWARD IS FORWARD metaphor - are explored in section 5.3.

4.7 Knowledge transfer and extension of domains

The succesful formation of metaphors will naturally introduce the possibility of transferring knowledge from one domain to the other. This is because the two environments are not completely isomorphic. For instance, the number of states in the two environments is not the same. Furthermore, we run different intra-domain mechanisms in the two different environments, effectively making the agent build up different knowledge in the two domains for it to transfer.

We have not made any explicit mechanism for transferring the knowledge, but once the metaphor between the domains is established, we take this knowledge (e.g. an association network developed in the Bin world) to be common knowledge for both environments.

4.8 Connecting the modules into one system

Overall, the system does not run like an autonomous agent. When it should look for analogies, how it schedules its overall tasks, and when to end one phase and move on to the next, are all aspects which are controlled by us as designers. However, it should be clear in the text when we have done non-trivial things - effectively acting as an external intelligent force; and when we have done trivial things - things which could have been automated given more time (e.g. parsing the output of HDTP).

4.9 Summary

We started out by giving a brief outline of the system before we moved on to a description of the two environments with which the agent interacts: the Walker world and the Bin world. We then stated the innate capacities of the agent, and

moved on to describe the associative mechanisms of *action schemata* and *Hebbian learning*. A description of how the agent actively explores the environments in a situated fashion was given, showing how the agent decides how to act, employing *memory hints* to form simple hypotheses it subsequently tests.

We described two different types of metaphors: inter-domain analogies (metaphors between the two environments), and intra-domain metaphors (metaphors formed within one environment). Finally, we described how we take the agent to transfer the knowledge implicitly, given the mechanisms above.

Chapter 5

Experiments and results

In this chapter, we describe a set of experiments that were run to test the mechanisms outlined above. Some of these experiments are in fact parts of how the robot develops, so they should be regarded as parts of an overall incremental process. This chapter will give us insight in how well the mechanisms for abstraction, analogy-making, etc., outlined in chapter 4 above, work. We start off by inspecting the performance of the abstraction to action clusters in the Walker world. After this, we test the Hebbian learning scheme in the Bin world. We then move on to an experiment to discover the symmetry of moving backward and forward in the Walker world domain by forming an analogy within the domain.

After these intra-domain experiments, we use the predicates which result from the action-schema abstraction process to form the analogy between the two domains. Finally, we try altering the Walker world slightly to see the effect on the analogy between the two domains.

For details on the implementation, we refer the reader to the appendix.

5.1 The first abstractions and associations

The first thing to test was the mechanism for abstraction to action clusters as described in sections 4.4.1 and 4.5. This mechanism has slighly different details in the two different domains, and will therefore be described in two different sections.

5.1.1 Action schemata in the Walker world

Recall that the agent clusters together *context-outcome* pairs based on an initial *memory hint* followed by a subsequent test of similarity. As noted before, the tests of similarity are always verified in the Walker world, simply because an action schema forms an equivalence class of action sequences, which implies transitivity of membership in a schema: for all action sequences A_1, A_2, A_3: if A_1 is in the same schema as A_2, and A_2 is in the same schema as A_3, then A_1 is in the same schema as A_3. In other words, as designers, we don't really have to test which equivalence classes will be formed - we already know this *a priori*.

We can give each action schema a name following a simple rule: for each action schema A, pick the shortest action sequence in it, S. Now, concatenate each single action in S, seperated by underscores. The result of this is the name of A. Using this naming convention, we can list all the action schemata, and their respective action clusters, in the Walker domain. This listing is shown in table 5.1.

Action schema	Action cluster
stay	(a,a), (b,b), (c,c), (d,d)
fwd	(a,b), (b,c), (c,d)
fwd_fwd	(a,c), (b,d)
fwd_fwd_fwd	(a,d)
bwd	(b,a), (c,b), (d,c)
bwd_bwd	(c,a), (d,b)
bwd_bwd_bwd	(d,a)

Table 5.1: Action schemata and clusters in the Walker world

After having done this clustering, the agent has essentially built a representation of the world based on its actions and observing the outcomes of its action. This representation reflects the underlying finite-state automaton structure of the Walker world which is shown in figure 5.1.

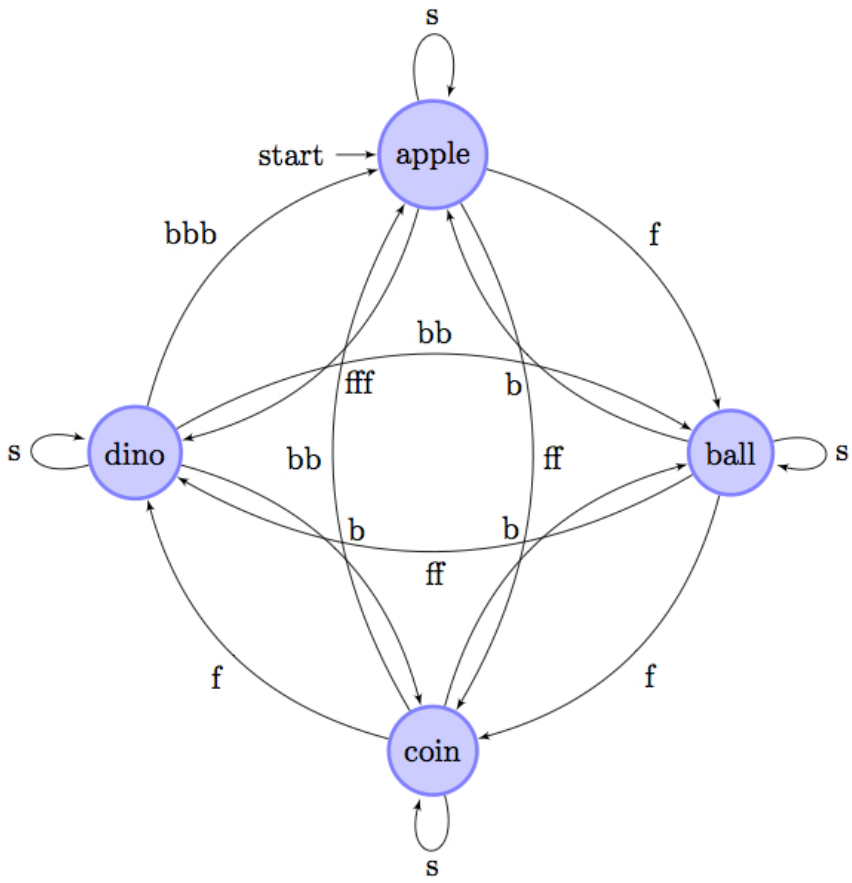

Figure 5.1: The finite-state automaton structure of the Walker world. Here the original landmark names are used for the nodes, and the edges are labeled with the action schema used to get from one state to another. We violate our own naming convention of predicates to make the graph more readable: *f* corresponds to *fwd*, *s* to *stay*, *b* to *bwd*, *ff* to *fwd_fwd*, and so on.

What we don't know, however, is how long the agent will spend trying to reach this segmentation of context-outcome pairs into the specified equivalence classes. Consequently, we checked how many action sequences the agent had to make before it had formed all the action schemas and clusters as listed in table 5.1. The result is shown as a histogram in figure 5.2.

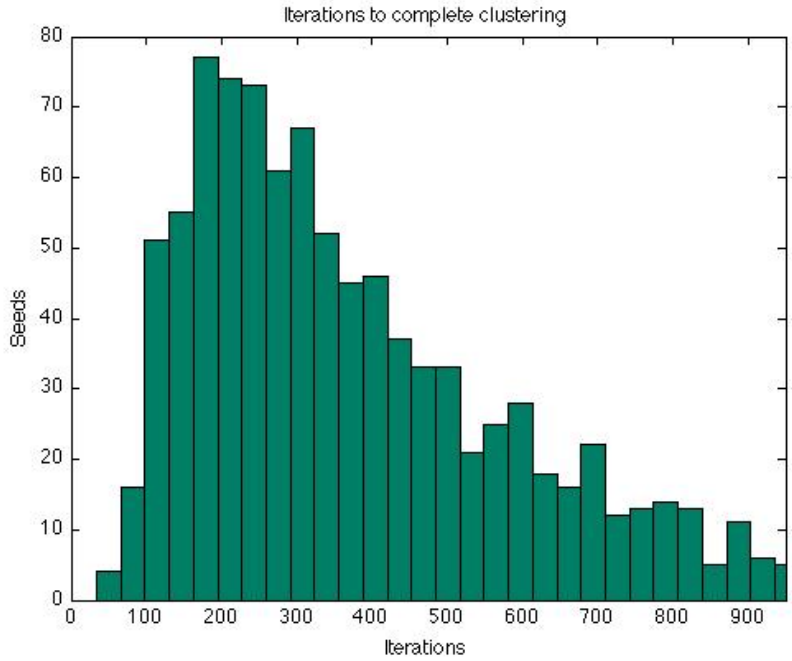

Figure 5.2: Distribution showing how many action sequences (iterations) the agent had to make in order to end up with a complete clustering. We could normalise this graph in order to make it a probability distribution. It would then represent the probability $y = P(x)$ of running for x iterations given seeds from a uniform distribution between 1 and 1000. Tested for 1000 different seeds.

5.1.2 Associations in the Bin world

The agent uses the same mechanism for abstraction to action schemata in the Bin world, and ends up with the set of schemata and clusters as shown in table 5.2.

Action schema	Action cluster
add	$(0,1)$, $(1,2)$, $(2,3)$, $(3,4)$
add_add	$(0,2)$, $(1,3)$, $(2,4)$
add_add_add	$(0,3)$, $(1,4)$
add_add_add_add	$(0,4)$
add_rem	$(0,0)$, $(1,1)$, $(2,2)$, $(3,3)$, $(4,4)$
rem	$(1,0)$, $(2,1)$, $(3,2)$, $(4,3)$
rem_rem	$(2,0)$, $(3,1)$, $(4,2)$
rem_rem_rem	$(3,0)$, $(4,1)$
rem_rem_rem_rem	$(4,0)$

Table 5.2: Action schemata and clusters in the Bin world. We have written the states as numbers representing the cardinality of the collection in that state. Of course, they could have arbitrary names, but using numbers makes the clusters more legible for the reader.

There are three things to note about this table, when we compare it to the Walker world one (table 5.1). First of all, there are more action schemata and clusters than in the Walker world. This is a direct result of there being more states in the Bin world.

Second, the agent does not have a dedicated "identity predicate", but forms one using *add* and *remove* put together. This is simply because it does not have the option of "staying" like it does in the Walker world.

Third, the "many" state is not included as a constant in any cluster. This is because the agent does not record *context-outcome* pairs containing the "many" state[1]. If it did, the clustering scheme would make all the action schemata collapse into one big action schema, because of the transitivity of clustering. A moment's thought will make the reader realise this[2] .

[1]Recall the interpretation of "many" as a "don't-care" state.

[2]If not: consider being in state "four" executing *add* or *add-add*. Both of these action sequences have the outcome "many", meaning that *add* and *add-add* (and their respective clusters) are abstracted to the same action schema. This forces, for instance, $(1,2)$ and $(1,3)$ into the same cluster. The same would happen if the robot executed *add-add-add* or *add-add-add-add*; and the analogous would happen for the *remove* sequences if the context were "many" and the outcome "four". Furthermore, the agent could execute *add* in the context "many" and achieve the outcome "many", meaning that all the identity clusters ($(1,1)$, $(2,2)$, etc.) would be clustered together with $(1,2)$, etc.

5.2 Hebbian association of observable states

We moved on to test the mechanism for associating events as described in section 4.4.2. We only implemented and tested the mechanism in the Bin world environment. Here, the events we refer to are the observable states in the world, which are the differently sized collections of blocks.

Since the agent only observes one state at a time, we have to do something else than base the learning on pure co-occurrence. Our solution to this problem is to let there be a time lag of one step, which means that co-occurrences are in fact transitions between states. The resulting network from letting the agent perform 2000 agent sequences, and have a learning rate $\eta = 0.01$ is shown in figure 5.3. A different representation of the same network is shown in figure 5.4. Note that the data has not been normalised, meaning that states which have been visited many times, have edges with higher weights than the average, and conversely for states which have not been visited many times.

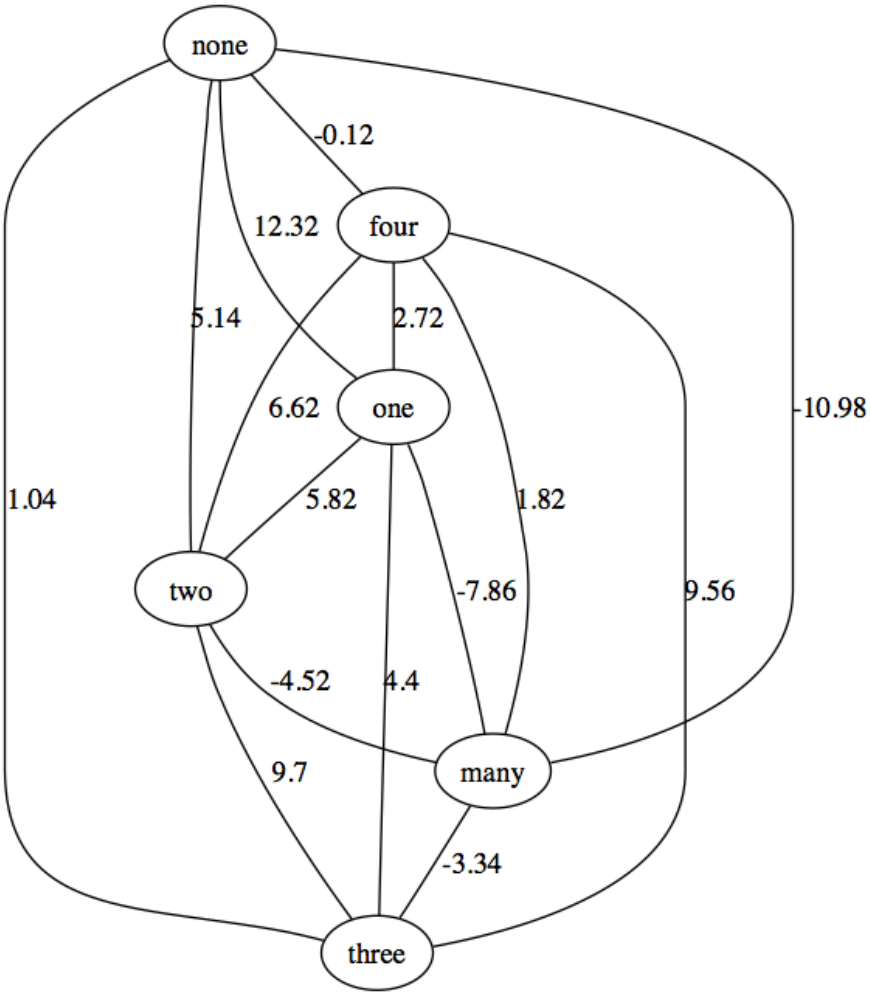

Figure 5.3: Example of association network where the nodes are observable states. This network was generated in the Bin world with the agent performing 2000 action sequences, and with a learning rate $\eta = 0.01$. The weight of an edge connected to a node N should be compared to the other edges connected to N. The weights show that collection states close together are associated with each other. For example, "one" has a high associational strength with "two" and "none", lower strength with "three" and "four", and a negative association with "many".

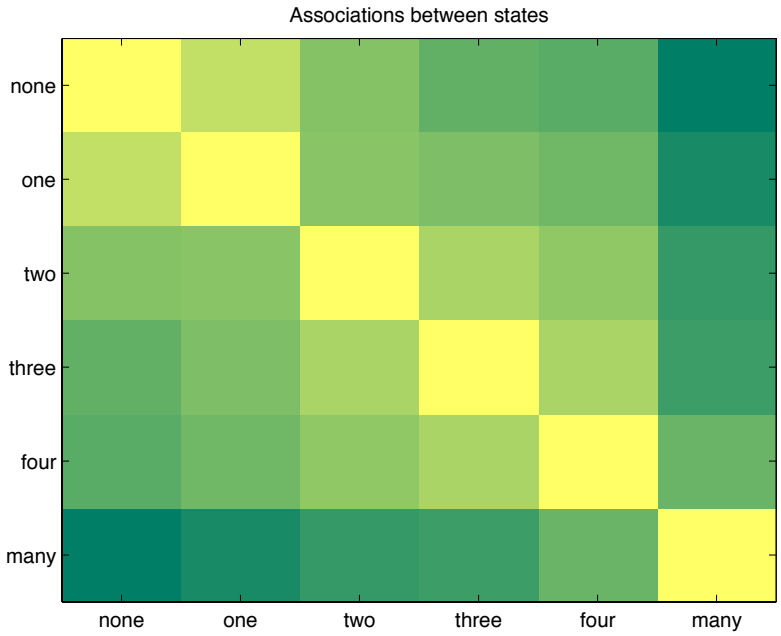

Figure 5.4: Alternative representation of association network. Yellow indicates strong indication, while green indicates weak (or inverse) association. The yellow diagonal represents the perfect association between every state and itself. We see that assocation strength is lowered when moving away from the yellow diagonal. This reflects what we can call a *principle of locality* in the Bin world: the agent is more likely to move to a close state than to a state far away, and therefore associates states close to each other with each other. Of course, this is just a reflection of how the action sequences are randomly generated.

We see that the agent associates states that are close to each other, where "close" means that you can get from one to the other using few (or one, or none) steps. This reflects the random generation of action sequences, where shorter sequences are more probable. We can refer to the concept associated with this resulting network as *state locality*. This is the concept of a set of states being local - or "close" - to other state. The concept is illustrated in figure 5.5.

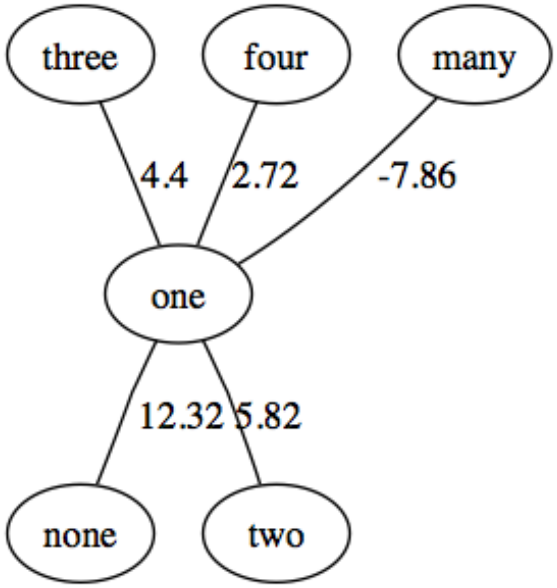

Figure 5.5: Subgraph showing the locality of “one”. State “none” and “two” have the strongest associational links, “three” and “four” have less, and “many” has a negative associational strength,

5.3 Symmetry by analogy

Having built the first basic concepts from action schemata, the agent now has a simple conceptual apparatus which is grounded in its exploration in the different worlds. This means that we can do new experiments based on this simple conceptual framework.

The first experiment done on the system’s metaphorising capacities was to take the clusters representing the action schemata which the agent had formed in the Walker world - such as *fwd*, *bwd*, *stay* - and try to form metaphors *within* the Walker domain. This is perhaps an unorthodox way of thinking about metaphors, but it is important to keep in mind that the world does not come readily segmented into domains, and this means that trying to form metaphors within what we as designers think of as a domain - having constructed it - is a perfectly reasonable thing for the system to do.

BACKWARD IS FORWARD

Source domain FORWARD		*Target domain* BACKWARD
fwd	$\rightarrow$	bwd
fwd_fwd	$\rightarrow$	bwd_bwd
fwd_fwd_fwd	$\rightarrow$	bwd_bwd_bwd
a	$\rightarrow$	d
b	$\rightarrow$	c
c	$\rightarrow$	b
d	$\rightarrow$	a

Figure 5.6: The BACKWARD IS FORWARD metaphor.

5.3.1 The BACKWARD IS FORWARD metaphor

The main analogy we expect the system to form here is a analogy between the *forward* predicates and the *backward* predicates. This is because these sets of predicates have the same internal structure of arguments. This structure is evident in the BACKWARD IS FORWARD metaphor, displayed in figure 5.6. As the reader probably has noticed, there is nothing inherent in these predicates which would imply that one is forward and the other is backward - these are labels we have conventionally given them because can correspond to our concept of forward and backward. However one only makes sense as a direction relative to the other, meaning they could just as well be *up* and *down*, or *left* and *right*.

This metaphor is exactly what we get as output from HDTP (with an evaluated total cost of 14) if we give it the input listed in figure 5.7.

Note that constants a, b, c, d are prefixed with s or t, depending on whether they belong to the source or to the target domain. This is done in order to prevent a naïve direct syntactical mapping between the constants in the source and the target domains.

However, splitting the domain up into these two smaller sub-domains is already doing half the job for the system. We want it to find the analogy without being spoon-fed the segmentation of the domain. This calls for a mechanism which makes the agent do the segmentation autonomously. We tried two different simple seg-

```
analogy('symmetry',
          domain('domain1',
          [
        fwd(s_a,s_b),
      fwd(s_b,s_c),
      fwd(s_c,s_d),
      fwd_fwd(s_a,s_c),
      fwd_fwd(s_b,s_d),
      fwd_fwd_fwd(s_a,s_d)
          ]),
          domain('domain1',
          [
        bwd(t_b,t_a),
      bwd(t_c,t_b),
      bwd(t_d,t_c),
      bwd_bwd(t_c,t_a),
      bwd_bwd(t_d,t_b),
      bwd_bwd_bwd(t_d,t_a)
          ])
).
```

Figure 5.7: Input to HDTP which results in the formation of the FORWARD IS BACKWARD metaphor.

mentation strategies: predicate-cluster isolation and non-singular segmentation.

5.3.2 Predicate-cluster isolation

The first segmentation mechanism we tried can be called *predicate-cluster isolation*, where a *predicate cluster* is just the set of statements involving a particular predicate (e.g. the predicate cluster of *fwd_fwd* is $\{fwd\_fwd(a,c), fwd\_fwd(b,d)\}$)[3]. Using this mechanism we simply isolate one cluster from the set of all clusters, and then check for analogies using HDTP between the isolated cluster and the rest of the clusters. Figure 5.8 shows an example for the isolation of the *bwd* cluster.

[3]Another way of thinking about this is that a predicate cluster is just an action cluster with its corresponding action schema as a predicate appended in front of each pair in the cluster.

```
analogy('symmetry',
        domain('domain1',
        [
    stay(s_a,s_a),
    stay(s_b,s_b),
    stay(s_c,s_c),
    stay(s_d,s_d),
    fwd(s_a,s_b),
    fwd(s_b,s_c),
    fwd(s_c,s_d),
    fwd_fwd(s_a,s_c),
    fwd_fwd(s_b,s_d),
    fwd_fwd_fwd(s_a,s_d),
    bwd(s_b,s_a),
    bwd(s_c,s_b),
    bwd(s_d,s_c),
    bwd_bwd(s_c,s_a),
    bwd_bwd(s_d,s_b),
    bwd_bwd_bwd(s_d,s_a)
        ]),
        domain('domain1',
          [
    bwd(t_b,t_a),
    bwd(t_c,t_b),
    bwd(t_d,t_c)
        ])
).
```

Figure 5.8: Example of input to HDTP for one isolated predicate cluster (*bwd*) in order to find analogies with other predicate clusters. Again, note that constants in the source and target domains must be renamed in order to prevent a naïve mapping.

In table 5.3 we see some statistics of the results for the predicate-cluster isolation scheme.

Predicate	Analogies formed ($\#A$)	Symmetry analogies formed ($\#SA$)	Symmetry percentage ($\#SA/\#A$)	Cost
stay	0	-	-	-
fwd	1	1	100%	70
fwd_fwd	4	2	50%	82
fwd_fwd_fwd	11	1	9.1%	90
bwd	1	1	100%	70
bwd_bwd	4	2	50%	82
bwd_bwd_bwd	11	1	9.1%	90

Table 5.3: Analogies and symmetry results. A "symmetry analogy" is an instance (or a non-empty sub-instance) of the BACKWARD IS FORWARD metaphor shown in figure 5.6. The "symmetry percentage" is the number of symmetry analogies formed divided by the total number of analogies formed. The results for *forward* and *backward* predicates are they same, since they are perfectly isomorphic. The cost is actually per analogy, but it was the same for all analogies given the same predicate.

We cannot use only the analogies produced by HDTP in isolation. For instance, in the case of *fwd_fwd_fwd,* 11 analogies are being formed, simply because the formula *fwd_fwd_fwd(t_a,t_d)* can be anti-unified with (and thus mapped to) each of the 11 formulae in the source domain where the two arguments are different (meaning all formulae except the 4 *stay* formulae). This is because its internal structure of the constants involved is the most basic (or least specific) of all the predicate clusters. This is illustrated in figure 5.9. The only mapping which occurs in all of the isolated-predicate attempts (except for *stay*, where no analogies are formed) is indeed the symmetry analogy, corresponding to the BACKWARD IS FORWARD metaphor.

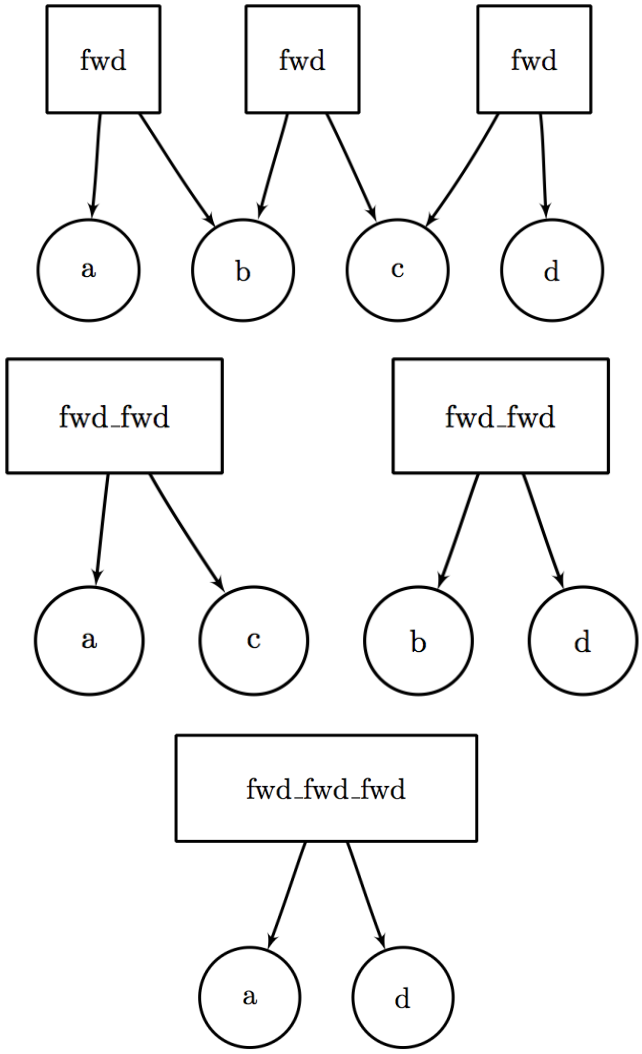

Figure 5.9: The structure of the *fwd*, *fwd_fwd* and *fwd_fwd_fwd* predicate clusters. Boxes denote predicates and circles denote constants. Note that the *fwd* predicate cluster is the only one capturing the internal structure which shows the connection between all the states in the world.

5.3.3 Non-singular segmentation

The second segmentation mechanism can be called *non-singular segmentation*, and is based on splitting the set of predicate clusters into two sets (i.e. domains) such that the minimum amount of clusters in each of the domains is two. This is

done to avoid the degenerate case of *fwd_fwd_fwd* described above for the predicate-cluster isolation scheme. The idea here is that by including more clusters in each domain, more of the internal structure of the world is being maintained, which then should lead to more interesting and useful intra-domain analogies. If a large enough portion of the segmentations consists mainly of the symmetry analogy, then the segmentation could be done randomly, using non-singularity as a heuristic.

For the experiment, we systematically tested all possible segmentations, to see the percentage of the analogies in which the symmetry analogy occured. For L predicate clusters, we can enumerate each cluster and represent a single source-target segmentation as a binary string of length L, where a 1 in position i denotes that cluster i is in the source domain and 0 means that it is in the target domain. Since we remove strings leading to singular and empty domains[4], we have the following formula[5] for the number of different segmentations, N:

$$N = 2^L - 2L - 2$$

Since we have seven predicates, we have that $N = 2^7 - 2 \times 7 - 2 = 112$ different segmentations. This number is too high for us to include the results for all segmentations. Instead, we summarise the results with the statistics shown in table 5.4.

Total number of analogies formed ($\#A$)	Total symmetry analogies formed ($\#SA$)	Total symmetry percentage ($\#SA/\#A$)	Average number of analogies per segmentation	Variance of the number of analogies per segmentation
1607	830	51.65%	14.35	166

Table 5.4: Table of results from non-singular segmentation. Note that in 4 cases, HDTP crashed because of a stack overflow. These cases are not included in the data, so we are really only dealing with 108 different segmentations.

[4] A singular domain occurs when a binary string has only a single 1 or a single 0 - meaning that there are $2L$ "singular" binary strings. An empty domain occurs when a binary string has only 1's or only 0's - which means that there are 2 "empty" strings.

[5] For $L > 2$.

The first thing we notice here is that the number of analogies being formed is very high. This is because as the number of predicates in the two domains gets higher, HDTP produces exponentially many more different analogies. Many of the different analogies can be said to be somewhat equivalent - or at least consistent with each other - having the same mapping of constants and predicates, but transferring different parts of the source domain to the target domain. We will come back to this in section 6.1.2, when discussing the results.

5.4 Bridging the metaphorical gap between the domains

The central metaphor of the project is of course the analogy between the Walker world and the Bin world - corresponding to the analogy between the two source domains from Lakoff and Núñez's grounding metaphors ARITHMETIC IS OBJECT COLLECTION and ARITHMETIC IS MOTION ALONG A PATH. By mapping the two source domains to each other, we obtain the metaphor MOTION ALONG A PATH IS OBJECT COLLECTION.

5.4.1 The MOTION ALONG A PATH IS OBJECT COLLECTION metaphor

The metaphor we expect the agent to form between the two domains is the one displayed in figure 5.10. There are a couple of variations of this metaphor. First of all, we could imagine the same metaphor, but with *add* being mapped to *fwd* and *rem* mapped to *bwd*. This is completely consistent with the two environments, as there is nothing which specifies direction in the Walker world. Indeed, this is precisely what gives rise to the BACKWARD IS FORWARD given above.

In the Bin world there is a slight difference between *add* and *rem* in that when the agent reaches the *none* state, it cannot keep removing things, while when the agent reaches the *many* state, it can keep adding things (up to a certain point). This difference is, however, only reflected in the transition probabilities, and because the agent ignores the *many* states while recording transitions, the difference between *rem* and *add* is not evident in the predicates abstracted from the Bin world.

MOTION ALONG A PATH IS OBJECT COLLECTION

Source domain THE BIN WORLD		*Target domain* THE WALKER WORLD
cardinality	→	landmark
add	→	fwd
add_add	→	fwd_fwd
add_add_add	→	fwd_fwd_fwd
rem	→	bwd
rem_rem	→	bwd_bwd
rem_rem_rem	→	bwd_bwd_bwd
$none$	→	$apple$
one	→	$ball$
two	→	$coin$
$three$	→	$dinosaur$

Figure 5.10: The MOTION ALONG A PATH IS OBJECT COLLECTION metaphor.

5.4.2 Running HDTP on the predicates built from associations

When we give HDTP the input listed in figure 5.11 which has been developed in the action-schema stage, the system forms two analogies. These two analogies are both instances of the MOTION ALONG A PATH IS OBJECT COLLECTION metaphor shown above, except that one of them is reversed. This means that *rem* and *add* are switched around, and consequently also the constants. This is shown in figure 5.12.

In addition to the mapping, the Walker world domain was also extended to include another object. This object is another landmark corresponding to the extra state in the Bin world. For the "reversed" mapping, this was a landmark corresponding to *none*, and for the "normal" mapping it was *four*. Furthermore, the correct predicate was introduced for this

Unfortunately, we were forced to make a selection of which predicates we could give HDTP as input. This was because of memory restrictions, as the current implementation of HDTP is not intended to take large amounts of predicates. Considering our earlier point that a complete analogy-making system should do all steps of the

```
analogy('walkiscol',
  domain('collection',
  [
collection(none),
collection(one),
collection(two),
collection(three),
collection(four),
add(none,one),
add(one,two),
add(three,four),
add(two,three),
rem(four,three),
rem(one,none),
rem(three,two),
rem(two,one)
  ]),
  domain('path',
  [
landmark(apple),
landmark(ball),
landmark(coin),
landmark(dinosaur),
bwd(ball,apple),
bwd(coin,ball),
bwd(dinosaur,coin),
fwd(apple,ball),
fwd(ball,coin),
fwd(coin,dinosaur),
stay(apple,apple),
stay(ball,ball),
stay(coin,coin),
stay(dinosaur,dinosaur)
  ])
).
```

Figure 5.11: Inter-domain input to HDTP.

MOTION ALONG A PATH IS OBJECT COLLECTION

Source domain THE BIN WORLD		*Target domain* THE WALKER WORLD
cardinality	→	landmark
rem	→	fwd
rem_rem	→	fwd_fwd
rem_rem_rem	→	fwd_fwd_fwd
add	→	bwd
add_add	→	bwd_bwd
add_add_add	→	bwd_bwd_bwd
$four$	→	$apple$
$three$	→	$ball$
two	→	$coin$
one	→	$dinosaur$

Figure 5.12: The "reversed" MOTION ALONG A PATH IS OBJECT COLLECTION metaphor.

process more or less autonomously, this is a bit of a methodological problem. However, there are several segmentation schemes which could reasonably have led to the input we gave HDTP in the end. Perhaps more importantly, if we had given HDTP the full input (i.e. the rest of the predicates as well), one would expect that the same two analogies be the output, seeing as the other predicates are completely consistent with the analogies resulting from the input in figure 5.11. The reason we used the input we did was that this included the predicates with the richest internal structure (e.g. *fwd* instead of *fwd_fwd*). Also, we only ran HDTP one way - that is, using the Bin world as the source domain. We could have done the opposite (or both), but chose to use the Bin world as the source domain because it had more predicates in total (and more constants), meaning that transfers were already more likely to go from the Bin world to the Walker world. Running HDTP with the domains shifted around would presumably entail the introduction of *stay* as a predicate into the Bin world.

5.5 Changing the environment

One of the interesting questions which naturally arise from the concept of an embodied mathematics is: if humans had a different body, or if lived in different environments, how would mathematics look? We do not attempt to answer this question here, but try a simple experiment to see how a different environment will changes the concepts developed if the agent still has the same capacities as before.

5.5.1 Making the Walker world loop

Recall that in the Walker world, when reaching an end state, there is no way of going beyond that end state. We can lift this restriction by making a loop: we connect the *apple* landmark directly to the *dinosaur* landmark so that moving backwards from *apple* takes the agent to *dinosaur*, and moving forwards from *dinosaur* takes the agent to *apple*. This changes the action schemata and their clusters as follows. Since the pair (a, d), which in the original world belonged to *fwd_fwd_fwd*, is added to the *bwd* schema, the *fwd_fwd_fwd* schema collapses into the *bwd* schema. Analogously, the *bwd_bwd_bwd* schema collapses into *fwd* because of the pair (d, a). Furthermore, since the pairs of *bwd_bwd* - i.e. the pairs (c, a) and (d, b) - both get added to *fwd_fwd*, and vice versa, *fwd_fwd* and *bwd_bwd* collapse into one single schema. We'll call this new schema *fwd_fwd*[6]. In table 5.5 we see the resulting new action schemata and clusters. Compare this to the original table 5.1.

Action schema	Action cluster
stay	(a, a), (b, b), (c, c), (d, d)
fwd	(a, b), (b, c), (c, d), (d, a)
fwd_fwd	(a, c), (b, d), (c, a), (d, b)
bwd	(a, d), (b, a), (c, b), (d, c)

Table 5.5: Action schemata and clusters in the looping Walker world

[6] Of course, it could also have been called *bwd_bwd*.

```
analogy('walkiscol',
  domain('collection',
  [
collection(none),
collection(one),
collection(two),
collection(three),
collection(four),
add(none,one),
add(one,two),
add(three,four),
add(two,three),
rem(four,three),
rem(one,none),
rem(three,two),
rem(two,one)
  ]),
  domain('path',
  [
landmark(apple),
landmark(ball),
landmark(coin),
landmark(dinosaur),
bwd(ball,apple),
bwd(coin,ball),
bwd(dinosaur,coin),
fwd(apple,ball),
fwd(ball,coin),
fwd(coin,dinosaur),
stay(apple,apple),
stay(ball,ball),
stay(coin,coin),
stay(dinosaur,dinosaur)
  ])
).
```

Figure 5.13: Looping Walker world input to HDTP.

Unlike the result in the previous experiment, this input causes a large number of analogies to be formed. In total, there are now 24 analogies formed, corresponding to various rotation of mappings which now become consistent because of the loop in the Walker world.

5.6 Summary

In this chapter we have seen a set of experiments which can be taken to illustrate the developmental progress of the agent. The first experiment showed how the agent builds up its first concepts (represented as action schemata) from its sensorimotoric primitives. The second experiment showed how the agent built up an association network between the different states (i.e. collection sizes) in the Bin world. In the third experiment we let the agent form intra-domain metaphors, resulting in the most prevalent of the metaphors, the BACKWARD IS FORWARD metaphor.

We then went on to the fourth experiment, which involved forming a metaphor between the two different environments. This experiment resulted in two different metaphors, both instances of the MOTION ALONG A PATH IS OBJECT COLLECTION. Finally, we performed a fifth experiment to see how the agent would form metaphors between two domains, one of the being the Walker world, and the other being an altered version of the Walker world. This resulted in more metaphors formed, with similar structures but no clear candidates like in the original experiment.

Chapter 6

Evaluation and discussion

In this chapter, we evaluate the results from the experiments carried out in the previous chapter. We discuss the the development of the mathematical concepts of state locality, symmetry, number, addition and subtraction. We also discuss Hofstadter's criticism of not addressing the problem of "extracting the gist" of a concept, and we briefly touch upon the distinction between metaphor and abstraction.

Again, for implementation details, we refer the reader to the appendix.

6.1 Which mathematical concepts have been developed?

There are two questions which must be answered when evaluating the system's performance. Firstly, do the results indicate the development of any *concepts*? And secondly, if so, can these concepts be characterised as *mathematical* concepts?

The main concepts which are in question are: *state locality* (from the experiment in section 5.2), *symmetry* (from section 5.3), and foundational mathematical concepts such as *number*, *addition* and *subtraction* (all from section 5.4). We go through each of these concepts in order and discuss whether or not we can attribute the agent knowledge of it.

6.1.1 State locality

The state-locality concept is very much a situated, embodied concept. It shows how the agent's internal configuration changes in different (embodied) states based on the situated exploration it has done to build to the associational network. For instance, being in the "two" state activates a certain pattern. We can again draw a parallel to neural activation, and our brains by using Lakoff/Johnson's characterisation of an embodied concept: "An embodied concept is a neural structure that is actually part of, or makes use of, the sensorimotor system of our brains" [25, p. 20].

If we think of each observable state in the environment as having a certain neural structure associated with it (say, through visual input), then when the agent moves to a certain state, this pattern is activated, which in turn activates other neural structures (corresponding to other states) proportional to the weights of the edge connected to it.

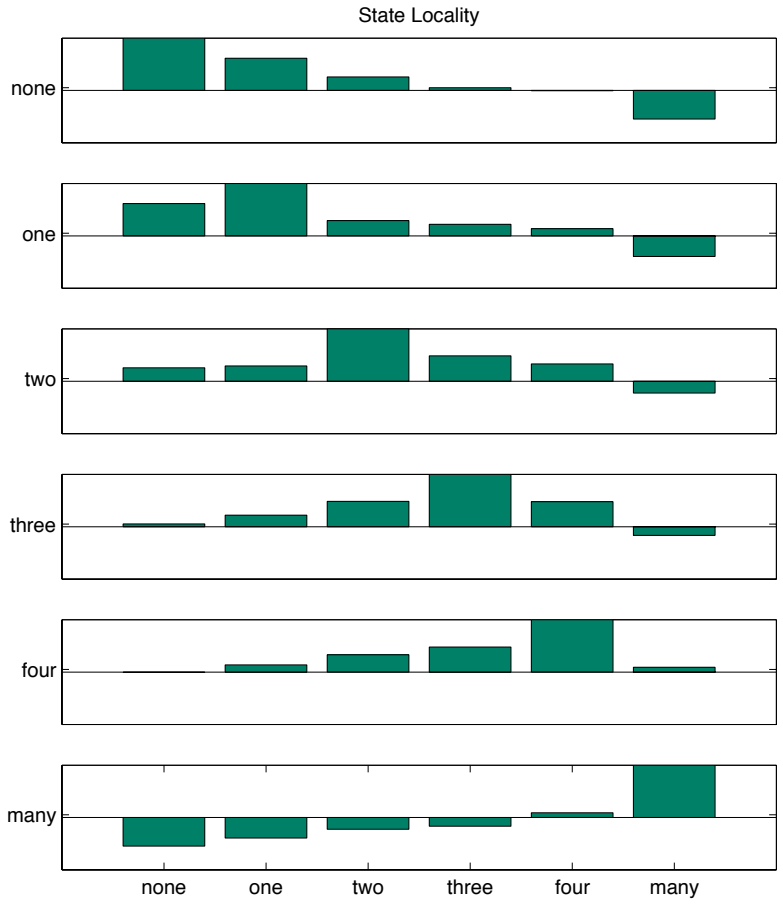

Figure 6.1: Figure showing state localities for the different states. Each subfigure shows the state locality for a particular state; for instance, the upper figure shows the state locality of "none". We can think of this as the "neural" activation of the agent's concepts of all the states when it is observing it is in the "none" state. The data used to generate this figure is the same as the data used in section 5.2. Again we point out that the data is not normalised with respect to frequency of transitions.

Figure 6.1 shows the "neural pattern" which results from being in each of the different states. Here, a neural pattern reads from left to right.

The state locality is exactly what gives the agent an embodied "meaning" associated the different observable states. This meaning is carried on upwards, as it develops the more abstract concept of a number, which builds on the observable

states in the environments it has explored. Thus, when the agent eventually maps a state to a number, it carries with it the embodied meaning of the state locality associated with the state, and now associates it with the number.

There are parallels here which could be drawn to findings in cognitive science. For instance, small numbers are associated with the left hand and large numbers with the right hand in human subjects - a phenomena known as the SNARC effect [10]. This is an example of embodied "meaning" reverberating up to the abstract concept of a number.

We conclude the discussion on state locality by saying that it is indeed a mathematical concept, mainly because of its relation to the mapping of a state to a number. After establising this mapping, the system gets a "feel" for the numbers - a "feel" which has been built up by its situated, embodied exploration, preliminary to the metaphoric mapping.

6.1.2 Symmetry

None of the two segmentation mechanisms described in 5.3 unambiguously indicated that the agent had developed the concept of symmetry. As an analogy, we can evaluate it based on the criteria mentioned in section 3.5; that is, we check if the system forms the expected analogy (in this case, the symmetry analogy), and if it only forms this analogy and not non-sensical ones.

Under both segmentation schemes, it forms the symmetry analogy for at least one particular segmentation. So it succeeds on the first criterion. The problem occurs because of the other analogies it forms. We already explained why this tends to happen for particular segmentations where the internal structure is too sparse (recall figure 5.9 which illustrates the problem).

Three questions arise, the first of which being: what are the other (non-symmetry) analogies which the system has formed? The answer to this is simply mappings such as *fwd* $\rightarrow$*bwd_bwd* and *bwd_bwd* $\rightarrow$*fwd_fwd_fwd* - mappings which do not make much sense. Of course, based on what HDTP is given as input, they make perfect sense, but that is the whole point: choosing what to give the analogy engine as input is half the job.

Secondly, in an analogy engine like HDTP, with several different outputs for

the same input, how do we assess which analogies should be accepted as "real analogies" - given the rest of the mechanisms of the system? The results from the non-singular segmentation scheme showed that, on average, HDTP produced more than 14 different analogies for each segmentation[1], all of which had the same, best cost evaluated by HDTP itself. Even though many of these analogies are very similar - in fact, some of them are the same analogy, but with e.g. different predicates transferred - the agent still has to commit to one analogy (or perhaps a few - but certainly not 200).

The third question is: how should this type of domain segmentation be done? Both our approaches can be described as brute-force mechanisms with certain restrictions on the domain (specifically, mechanisms with restrictions on the cardinality of the source and target domains). Perhaps the segmentation can be done in an intelligent way? For instance using some measure of similarity of predicates to partition them into two domains.

One other approach to evaluating the symmetry analogy is to assume that one analogy will be formed by the agent, and then pick the analogy which is most prevalent out of all the analogies formed. We could draw a meta-analogy[2] here and compare this to the way political parties are chosen in a democracy. We consider a set of parties/analogies, and the one with the most votes/occurrences is elected/adapted as the ruling-party/conceptual-metaphor [3].

Under this "democratic election of analogies", the symmetry analogy comes out on top in both segmentation schemes. It achieves an overall 51.65% in the non-singular segmentation scheme, which means we don't even have to check how many "votes" the "runner-up" receives. In the predicate-cluster isolation scheme, it has an overall 25%, but as there are no other analogies with more occurrences, it comes out as the winning analogy in this scheme as well.

An alternative to this straightforward percentage calculation would be to weight differently the segmentations which seem more "certain". For instance, in the *fwd* segmentation of the predicate-cluster isolation scheme, only one analogy is formed,

[1] Albeit with a very high variance, meaning that some segmentations produced more than 200 analogies, and others produces only 1 analogy.

[2] Or meta-metaphor.

[3] We will not speculate too much in the possible transfers latent in this meta-analogy, which might include such concepts as "coalition analogies" and "conceptual overthrows".

namely the symmetry analogy. The system could then weight this analogy higher than each of the 11 analogies formed in the *fwd_fwd_fwd* segmentation.

This suggested approach is vaguely reminiscent to a set of machine learning techniques called *ensemble methods*, where a set of models are trained on different segments of a data set, and then somehow combined to yield better overall predictive performance. To strengthen our "analogy election" metaphor, we can point out that the different models in ensemble methods often are called "committees" [1].

The "democratic election of analogies" introduces another problem. Recall that the assumption is that every time the system tries to form analogies, it has to pick one. Of course, sometimes there is no reasonable analogy to form between two domains, in which case no analogy should be picked. A simple way to fix this would be to only pick an analogy if it is above a certain threshold.

To summarise, on the one hand, under our original criteria for analogies, the symmetry analogy fails on criteria 2 and 3, which means that we could argue that it does not develop a concept of symmetry. On the other hand, it shows a clear preference for the symmetry analogy over any other analogy under both the non-singular segmentation scheme and the predicate-cluster isolation scheme.

6.1.3 Number, addition and subtraction

The concept of a number emerges through the anti-unification of the predicates *cardinality* and *landmark*. These two predicates each represent the source domain equivalent of a number in Lakoff and Núñez's grounding metaphors ARITHMETIC IS OBJECT COLLECTION and ARITHMETIC IS MOTION ALONG A PATH respectively. One interesting result is that the system has no preference to map *fwd* to *add* and *bwd* to *rem* versus mapping *fwd* to *rem* and *bwd* to *add*. One the one hand, this can be said to be an obvious result because there is nothing to distinguish forward from backward, but on the other hand this shows that the mapping of 'adding blocks to a collection' to 'moving forward' is a completely arbitrary choice! It could just as well have been the other way around.

This leads us, then, to the question: why do we as humans associate the forward direction with adding blocks? One possible answer is that we already have even more fundamental underlying metaphors in our conceptual system which force us

to map 'forward' to 'add' in order to keep this conceptual system coherent. This is precisely what Lakoff and Johnson [24] argue. In fact, the general consistency among humans to map 'forward' to 'add' could be taken as evidence of their hypothesis that conceptual metaphors interact in a coherent manner with more basic metaphors such as MORE IS UP and UP IS FORWARD.

Addition and subtraction are thought to be general operators, and the agent does not have any mechanism which would allow it to see the relationship between *fwd*, *fwd_fwd* and *fwd_fwd_fwd* (and similarly for the *bwd* predicates). The fact that they are all instances of the more general addition (subtraction) operation lies at a higher level of abstraction. This type of ongoing higher levels of abstraction is of course a very important part of mathematics. We would say that for any agent to be called truly mathematically competent, it should have mechanisms which would allow it to perform this sort of recursive cognition - what Hofstadter [19] calls "strange loops". However, our agent still can be said to have developed some preliminary concepts of addition and subtraction.

Our last experiment with the looping Walker world showed that when the structural difference between two domains becomes bigger, it can have the effect that more analogies are formed. This can presumably make it harder for the system to commit to one single analogy, since it has too many possibilities to choose from. Thus, our system did not form a "new mathematics" using the altered Walker world, but rather become less certain of the metaphor between the two domains.

6.2 Addressing Hofstadter's criticism: is it "extracting the gist"?

Hofstadter [18] has criticised analogy researchers for not studying all stages of analogy-making - including representation and elaboration - but only focusing on the mapping stage. Forming such representations and deciding which parts of a domain are relevant is what Hofstadter calls "extracting the gist" of a concept. We have tried to show that our system does develop a representation in a more or less unsupervised fashion, meaning that we do not commit the fallacy of simply giving HDTP a set of predicates which it can then perform the mapping step on.

We saw that not only is representing statements in a domain, and knowing which

parts are relevant to a metaphor, important to perform successful analogical reasoning - sometimes just knowing just what to represent as its own domain can be a challenge (cf. our problem of intra-domain segmentation with the symmetry analogy).

The way in which Hofstadter's criticism mainly relates to our project is in how we have set up the two environments. One could argue that by setting up the two environments in such a simple manner, we have effectively "extracted the gist" before the agent even does anything. On the one hand, it is difficult to counter this argument, because there are of course no collections of blocks nor any landmarks as parts of the overall system. On the other, if we choose to approach the problem of developing mathematical concepts in a simulated world, there is no way out of this. Sticking to the simulated approach, what we could do is to make two environments which are realistic enough to resemble their real-world counterparts, and also give the agent a more generalised mechanism for formulating its first basic concepts.

To summarise, our approach to forming analogies does not go into the trap of spoon-feeding an analogy engine with a set of predicates. The predicates our agent uses are developed from its associative mechanisms and explorative behaviour. In this way, the agent does indeed "extract the gist" of its environments. However, whether or not the environments are too simple for there to be any "gist" there to extract, is arguable.

6.3 Metaphor or simply abstraction?

One criticism which has been raised against Lakoff's theory of metaphor, is that it over-uses the term "metaphor" in a way such that it applies to almost all types of abstraction and categorisation. Pinker [35] writes: "No one has a problem with the idea that the lens of an eye and the lens of a telescope are two instances of the general category 'lens', rather than the telescope being a 'metaphor' for the eye". Pinker's point here is that metaphor is being used unneccesarily to describe something which is just abstraction.

Interestingly, this debate is effectively neutralised by using HDTP, since in HDTP a metaphor is simply two instantiations anti-unified by a single general model. The

more general theory which results from anti-unifying the formulae in the source and target domains in HDTP can be thought of as a type of abstraction. Of course, it is a very specific type of abstraction, but in the case of Pinker's lens example, it is conceivable that running HDTP on an "eye domain" and a "telescope domain" would result in a model where the anti-unified "lens" predicate corresponds to the general concept of a lens.

In brief, HDTP allows us to connect "conceptual metaphor" to "concept" in a way that other analogy engines cannot. This is a strong point for using HDTP for this kind of approach to conceptual development.

6.4 Summary

We have seen that the results indicate that our agent has developed some sense of state locality, which is an embodied concept. Under some reasonable evaluation criteria we can say that the agent developed the concept of symmetry, but it fails on two of our original criteria since it also produces a range of non-sensical intra-domain metaphors. The inter-domain metaphor MOTION ALONG A PATH IS OBJECT COLLECTION shows that the agent can be said to have developed a concept of number, with very restricted versions of addition and subtraction.

Chapter 7

Conclusion and future work

7.1 Conclusion

The overall outcome of the project is difficult to judge, mainly because there are underlying philosophical issues for many of the topics we touch upon. By staying true to our original hypothesis, and the terminology involved in it, we can, however, say that we have shown something.

What we have shown in this project is that a mechanism for association (action schema clustering) allows an agent to build a simple representation (the predicates), which effectively enables it to start forming metaphors, both between clearly distinguishable environments and within the same environment. These metaphors represent what we can call basic mathematical concepts such as number and symmetry, as they are not only mappings between the domains, but represent a more general model given to us by heuristic-driven theory projection which unifies the two environments.

Furthermore, the Hebbian learning mechanism for associating states which each other gives the agent a situated, embodied "feel" for the states, which it in turn brings with it when the mapping onto the abstract domain of mathematics (represented by the model unifying the two environment predicates) is established.

Below, we give some suggestions for future work related to embodiment, mathematical cognition and metaphors. The field of mathematical cognition is very much a field yet to be explored, and we hope that researchers from artificial intelligence

will continue to contribute with computational approaches.

7.2 Suggestions for future work

The most obvious suggestion for future work, is bringing the agent into the real, physical world. Doing this would make any progress done on the development of mathematical concepts far less speculative. It would also require actually implementing mechanisms for object recognition, information gathering, curiosity drive, etc., which would make the overall system complete as an embodied, conceptualising agent. Clearly, it would also pose other challenges which would be less related to mathematical cognition directly, but which when dealt with, would give the results an overall validity which it would be harder to argue against.

Another suggestion for further research is to develop mechanisms for "moving beyond the explorable" - that is, mechanisms for generalisation and induction. These would start to touch upon more abstract parts of mathematics, and in that context we should also mention that further work on the *linking metaphors of mathematics* [26] which connect abstract mathematical concepts is another interesting direction of research.

As for research on analogies and metaphors, further exploration of how to categorise and segment different domains, and also using a sort of probabilistic approach to analogies as hinted at in section 6.1.2 could yield interesting results. It would also be interesting to see how the costs of HDTP could relate to a possible similar cost scheme in humans. Even though it was not developed to be psychologically realistic, there could still be interesting parallels between a logical framework of analogies and human analogy formation.

Finally, using association networks for embodied concepts seems like a promising direction in the study of embodiment in artificial intelligence. We think our simple Hebbian-learning scheme could have applications outside of mathematical cognition, and that it might also have a certain cognitive plausibility which could make it useful also in modelling human concept development.

Appendix A

Implementation details

Here we give some details on the implementation of the system. This includes how the Walker world and the Bin world were implemented, how we interacted with HDTP and how some of the experiments were carried out. Most of the system was implemented in Python, although we used Matlab, Graphviz and dot2texi to make graphs and figures.

A.1 Walker world and Bin world

Both the Walker world and the Bin world were implemented in Python. A state was represented by an integer, actions were operations on this integer (e.g. *fwd* was adding 1 to the state integer) and observing the state was simply checking what the integer was. In the Walker world, there was a string representing each of the landmarks, and in the Bin world there was a string representing each of the collection cardinalities.

The associative memory was represented by a dictionary data structure which contained items of a class called Entry. The entry class would have a cluster ID which made it possible to do clustering for developing the action schemata and clusters. The clustering itself was done in a recursive fashion so that the transitivity requirement of equivalence classes was maintained.

The overall running of the Walker world agent behaviour (which is similar to the Bin world agent behaviour) is seen in the following Python snippet:

```
schema = rnd_schema()
f = observe()
schema = execute(schema)
t = observe()
if not (f,t) in memory.keys():
    memory[(f,t)] = Entry(f,t)
memory[(f,t)].add_schema(schema)

# performing the schema 'activates' other stored entries
# which have used that schema - the 'similars'
similars = [v for k,v in memory.items() \
    if v.has_schema(schema) and not (v.s1,v.s2)==(f,t)]
more_than_one = [s for s in similars \
    if filter(lambda x:not x in memory[(f,t)].schemas,s.
       schemas)!=[]]
if more_than_one!=[]:
    rp = random.choice(more_than_one)
    h_schema = random.choice(\
                filter(lambda x:not x in memory[(f,t)].
                   schemas,rp.schemas))
    # do we have (t,f) in memory? if yes, then execute a
       schema
    # to get back to f
    if (t,f) in memory.keys():
        execute(memory[(t,f)].schemas[0])
        execute(h_schema)
        if landmarks[state]==t:
            memory[(f,t)].agglomerate(memory[(rp.s1,rp.s2)])
```

This shows how clustering, memory hints and storing of new memory entries is done.

A.2 Use of HDTP

We interacted with HDTP mainly through forming generating input strings in Python based on the predicates, etc., and then give this input to HDTP through a Python module we developed as an interface to HDTP, so we could try several different inputs in an automatic fashion. The output of HDTP was recorded in a log file which we could then inspect later. For example, the following function would generate input to HDTP given two sets of predicates (source and target domains):

```
1  def domains_to_hdtp(source, target, name='output'):
2      """
3      Takes two list of strings (expressions), and
4      returns a string which is of the HDTP analogy format.
5      """
6      res = "analogy('%(dom)s',\n\tdomain('domain1',\n\t[\n\t
           " % {'dom':name}
7      res += ",\n\t  ".join([ch_args(lambda s:'s_'+s,x) for x
          in source]) + "\n\t]),\n\t"
8      res += "domain('domain1',\n\t[\n\t  " % {'dom':name}
9      res += ",\n\t  ".join([ch_args(lambda s:'t_'+s,x) for x
          in target]) + "\n\t])\n).\n\t"
10     return res
```

Then the following function would run HDTP: first it converts its arguments into HDTP input (using the function above), and then it actually runs HDTP with this input before it writes the output to a result file:

```
1  def analogies_to_file(source, target, name='output'):
2      analogy = domains_to_hdtp(source,target,name)
3      tmp_path = '../hdtp/analogies/'+name+'.analogy'
4      f = open(tmp_path,'w')
5      f.write(analogy)
6      f.close()
7      out_path = '../data/results.dat'
8      results = open(out_path,'a')
```

```
9       # change dir to run hdtp
10      os.chdir('../hdtp/')
11      os.system('/opt/local/bin/swipl -c com.pl >> '+out_path)
12      results.close()
```

A.3 Experiments

A.3.1 Hebbian association network

The learning of the weights in the network was done simply by using the equation in 4.4.2. We did not compute the weights in an on-line fashion, but rather stored the concept activations in a matrix, and then computed the weights after a complete run. This is of course less realistic than doing it on-line, but the mathematics of it is the same. This simple code snippet shows how we use the differential equation to update the weights after a run of 2000 iterations:

```
1 weights = n.matrix([0]*len(names)) # initialise all weights
    to 0
2 activity = n.matrix(activity).transpose() # convert concept
    activations into matrix
3 for t in range(2000):
4   weights = weights + l_rate*activity[:,t]*activity[:,t].
      transpose()
```

A.3.2 Predicate-cluster isolation

Predicate-cluster isolation was done as follows:

```
1 def predicate_cluster_isolation():
2     """
3     Goes through all isolated-cluster combinations, looking
        for
4     analogies. Outputs the results in ../data/
        analogy_results.dat
```

```
5       """
6       pred_clusters = get_preds()
7       for i in range(len(pred_clusters)):
8           tmp = pred_clusters[0:i]+pred_clusters[i+1:] #
                remove isolated cluster
9           all = reduce(lambda x,y:x+y,tmp) # put all remaining
                predicates into one list
10          hdtp_interface.analogies_to_file(all,pred_clusters[i
                ]) # find analogies
```

A.3.3 Non-singular segmentation

The function below shows how non-singular segmentation was done.

```
1  def non_singular_segmentation():
2       """
3       Goes through all isolated-cluster combinations, looking
            for
4       analogies. Outputs the results in ../data/
            analogy_results.dat
5       """
6       pred_clusters = get_preds()
7       p = len(pred_clusters)
8
9       # generate binary strings:
10      bstr = [bin(x)[2:].zfill(p) for x in range(2**p)]
11      # then remove singular segmentation schemes:
12      bstr = filter(lambda x: 1 < sum([int(s) for s in x]) < p
            -1,bstr)
13      for s in bstr:
14          source = []
15          target = []
16          for i,c in enumerate(s):
```

```
17              if c=='1':
18                  source += pred_clusters[i]
19              else:
20                  target += pred_clusters[i]
21          hdtp_interface.analogies_to_file(source,target) #
               find analogies
```

Bibliography

[1] C.M. Bishop. *Pattern recognition and machine learning*. Springer New York:, 2006.

[2] P. Boero, L. Bazzini, and R. Garuti. Metaphors in teaching and learning mathematics: a case study concerning inequalities. In *Proceedings of the 25th International Conference, Psychology of Mathematics Education, July 11- 17, 2001*, volume 2, pages 185–192, Utrecht, The Netherlands, 2001.

[3] R.A. Brooks. Intelligence without reason. *Artificial intelligence: critical concepts*, 3, 1991.

[4] N. Chomsky. *Syntactic structures*. Walter de Gruyter, 2002.

[5] P. Dayan and L.F. Abbott. *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. MIT press Cambridge, MA:, 2001.

[6] S. Dehaene. *The number sense: How the mind creates mathematics*. Oxford University Press, USA, 1999.

[7] G.L. Drescher. *Made-up minds: a constructivist approach to artificial intelligence*. The MIT Press, 1991.

[8] B. Falkenhainer and K.D. Forbus Dedre. The structure-mapping engine: Algorithm and examples. *Artificial intelligence*, 41(1):1–63, 1989.

[9] J.A. Feldman. *From molecule to metaphor*. MIT Press Cambridge, MA:, 2006.

[10] W. Fias. The importance of magnitude information in numerical processing: Evidence from the SNARC effect. *Mathematical cognition*, 2(1):95–110, 1996.

[11] D. Gentner. Structure-mapping: A theoretical framework for analogy*. *Cognitive science*, 7(2):155–170, 1983.

[12] D. Gentner. The mechanisms of analogical reasoning. *Similarity and analogical reasoning*, pages 199–241, 1989.

[13] D. Gentner and A. Markman. Structure mapping in analogy and similarity. *Am.Psy.*, 52:48, 1997.

[14] A. Goldfain. Embodied enumeration: Appealing to activities for mathematical explanation. *Cognitive Robotics: Papers from CogRob2006*, 2006.

[15] J.E. Grady. Foundations of meaning: Primary metaphors and primary scenes. 1998.

[16] M. Guhe, A. Pease, and A. Smaill. A cognitive model of discovering commutativity. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pages 727–732, 2009.

[17] H. Gust, K.U. Kühnberger, and U. Schmid. Metaphors and heuristic-driven theory projection (HDTP). *Theoretical Computer Science*, 354(1):98–117, 2006.

[18] D. Hofstadter. A review of mental leaps: Analogy in creative thought. *AI Magazine*, 16(3):75–80.

[19] D.R. Hofstadter and E. Godel. *Bach: an eternal golden braid*. Basic Books, New York, 1979.

[20] D.R. Hofstadter and M. Mitchell. The Copycat Project: A model of mental fluidity and analogy-making. *Advances in connectionist and neural computation theory*, 2:31–112, 1994.

[21] E.T. Jaynes. *Probability theory: the logic of science*. Cambridge Univ Pr, 2003.

[22] I. Kant. Prolegomena to any Future Metaphysics, trans. *Paul Carus and revised by james W. ellington, Indianapolis, Hackett*, 1977.

[23] T. Kinn. *Pseudopartitives in Norwegian*. PhD thesis, University of Bergen, 2001.

[24] G. Lakoff and M. Johnson. *Metaphors we live by*. Chicago London, 1980.

[25] G. Lakoff and M. Johnson. *Philosophy in the Flesh*. Basic Books New York, 1999.

[26] G. Lakoff and R.E. Nunez. *Where mathematics comes from: how the embodied mind brings mathematics into being*. Basic Books, New York, 2000.

[27] S.M. Lamb. *Pathways of the brain: The neurocognitive basis of language*. J. Benjamins, 1998.

[28] E. Mowat and B. Davis. Interpreting embodied mathematics using network theory: Implications for mathematics education. *Complicity: An International Journal of Complexity and Education*, 7(1):1–31, 2010.

[29] R. Munroe. Analogies, August 2010. http://xkcd.com/762/.

[30] C. Murphy. The constructive role of the conceptual metaphor in children's arithmetic: A comparison and contrast of Piagetian and embodied learning perspectives. *Philosophy of Mathematics Education Journal*, 22, November 2007.

[31] S.S. Narayanan. *KARMA: Knowledge-based active representations for metaphor and aspect*. PhD thesis, UNIVERSITY of CALIFORNIA, 1997.

[32] J. Piaget. *The Child's Conception of Number*. Routledge & Kegan Paul Ltd, 1941.

[33] J. Piaget. *Biology and Knowledge: An Essay on the Relations between Organic Regulations and Cognitive Processes*. Edinburgh University Press, 1967.

[34] D. Pimm. Metaphor and analogy in mathematics. *For the Learning of Mathematics*, 1(3):47–50, 1981.

[35] S. Pinker. *The stuff of thought: Language as a window into human nature*. Viking Pr, 2007.

[36] G. Polya. *Mathematics and plausible reasoning: Induction and analogy in mathematics*. Princeton Univ Pr, 1990.

[37] S.K.R. Popper. *The logic of scientific discovery*. Hutchinson, 1959.

[38] T. Regier. *The human semantic potential: Spatial language and constrained connectionism*. The MIT Press, 1996.

[39] D. Schlimm. *Axiomatics as Engine for Driving Discovery in Mathematics and Science*. PhD thesis, Carnegie Mellon University, 2005.

[40] M. Schmidt. *Restricted Higher-Order Anti-Unification for Heuristic-Driven Theory Projection*. BSc thesis, University of Osnabrück, 2008.

[41] A. Schwering, U. Krumnack, K.U. Kühnberger, and H. Gust. Analogical reasoning with SMT and HDTP. In *2nd European Cognitive Science Conference (EuroCogSci07), Delphi, Greece*, pages 652–657, 2007.

[42] A. Schwering, U. Krumnack, K.U. Kühnberger, and H. Gust. Syntactic principles of heuristic-driven theory projection. *Cognitive Systems Research 10, 251-269*, 2009.

[43] A. Sfard. Reification as the birth of metaphor. *For the Learning of Mathematics*, 14(1):44–55, 1994.

[44] A. Sfard. On acquisition metaphor and participation metaphor for mathematics learning. In C. Alsina, J. M. Alvarez, B. Hodgson, V. Laborde, and A. Pérez, editors, *8th International Congress on Mathematical Education. Selected Lectures*, pages 397–411. Seville, Spain: S.A.E.M. Thales, 1996.

[45] M. Shanahan. Perception as abduction: Turning sensor data into meaningful representation. *Cognitive Science: A Multidisciplinary Journal*, 29(1):103–134, 2005.