Direct Probabilistic Semantics

A Contextualist Formal Semantic Model

master thesis Cognitive Artificial Intelligence
Universiteit Utrecht
45 ECTS

Jan Kanis

student number
0339954

Tuesday 27th August, 2013

primary supervisor:
Albert Visser

secondary supervisors:
Vincent van Oostrom
Rick Nouwen

Abstract

This thesis should be viewed in the context of the debate that is currently going on in the field of philosophy of language: the debate on the question of how much influence the context in which a sentence is used has on the meaning of that sentence. Closely related to this is the question of whether a sentence can have a meaning on its own, or if a sentence only has a meaning within the context and circumstances in which it is used.

Simplifying a bit, two approaches can be distinguished: that of literalism and of contextualism. Literalism tries to understand the meaning of language by representing it as a formula in a symbolic logic. Word meanings are taken from a lexicon as logical formulas, which are then composed according to the syntactic structure of a sentence to form the meaning of the whole sentence, which is a larger logical formula. Literalists include several mechanisms to introduce contextual information into this process, but according to contextualists these mechanisms are too limited. According to the more extreme contextualists, a sentence can be gotten to mean (almost) anything if it is used in the right context or situation.

I find that the arguments the contextualists make are generally convincing, and that a more contextual approach is needed. However one of the main drawbacks of contextualism is that the theories it proposes are not very well developed in a formal sense. This debate takes place mostly in the field of philosophy of language, and the theories proposed by contextualists are also usually formulated in philosophical terms and at a philosophical level of abstraction. On the other hand, literalists have models that are better developed and formalized well enough to program into a computer so that calculations and predictions can be made about them. Contextualist models usually do not reach this level of formal development, and it is therefore no surprise that in the field of formal linguistics the theories promoted by contextualists do not find a large audience.

In this thesis I aim to take a stab at this shortcoming. I have developed a model of semantics that is based on the ideas defended by contextualism, and implemented it in a computer program. An important property in which the model differs from many existing theories is that it is a non-truthconditional theory of meaning, and instead implements a more internalist approach. This does not prevent formalization and implementation in a computer, and in fact helps to answer some common philosophical problems. The resulting computer program has the ability to interpret and act upon simple sentences that are about a small scene containing a few geometric objects.

Contents

Chapter 1

Introduction

In this master thesis I will address the problem of formally developing a contextualistic model of the meaning of language.

Contextualism is one of two classes of positions in a debate in the philosophy of language, the other class being 'literalism'. Both of these terms are used with different meanings in other fields of linguistic and outside of linguistics, but I will only use them as they are used in the field of philosophy of language.[1]

Literalism and contextualism are two broad categories, and more detailed positions can be identified within them. In the interest of brevity I must simplify in the here following description of these positions.

Literalism generally tries to uncover the truth conditions of language[2] by representing the meaning of sentences as a formula in a symbolic logic. Word meanings are taken from a lexicon as logical formulas, which are then composed according to the syntactic structure of a sentence to form the meaning of the whole sentence, which is a larger logical formula. Since not all information on which the meaning of a sentence depends is always made explicit within a sentence, literalists propose several mechanisms to capture this contextual information. But according to contextualists, these proposed mechanisms are too limited. Contextualists hold that it is not possible to assign truth conditions to *sentences*, but only to a sentence's *use* in a specific context. Contextualism is a varied class of positions that have a common goal of allowing more contextual influences to play a role in the construction of the meanings of sentences than the methods that literalism recognizes. According to the more extreme contextualists, constructing something like a sentence meaning separately from the context in which the sentence is used is useless because a sentence can be gotten to mean (almost) anything if it is used in the right context or situation.

I find that the arguments the contextualists make are generally convinc-

[1]Specifically, I will follow the usage of [Recanati, 2004].

[2] [Recanati, 2004, p. 3]

ing, and that a more contextual approach is needed. However one of the main drawbacks of contextualism is that the theories it proposes are not very well developed in a formal sense. In this thesis I aim to take a stab at this shortcoming. I have developed a model that describes what the meanings of sentences in natural language are, and what kinds of contextual information is needed to obtain them. To be sure that the proposed model can be formalized without running into insurmountable problems, I have written a proof of concept implementation of the proposed model as a computer program.

In this thesis I will discuss the positions that make up the contestants in the debate and I will discuss the most important arguments. Then I will describe the proposed model and its computer implementation and discuss whether it can be used as a contextualist model of language meaning. Finally I will talk about ways to extend both the program and the theory, and then come to a conclusion.

1.1 Status Quaestionis

The current status of contextualist theory is that there is an active debate going on in the field of philosophy of language with opponents of the approach.[3] Both contextualist and non-contextualist positions are being actively defended and neither contextualism nor literalism can be said to be a clearly insignificant minority position. There are a number of theoretical proposals from the contextualist side, but the position is diverse as there can be large differences between different proposals that can be classified as contextualist.[4]

As far as I am aware none of the existing proposals is developed well enough in a formal sense that they could be implemented on a computer. The implementation on a computer is not a primary goal in itself from a theoretical point of view, but the lack of a good formalization brings with it a lot of vagueness in the discussion, up to the point where contextualists are being accused of being internally inconsistent[5]. The lack of good formally developed models also hinders the adoption of contextualist approaches in

[3]Some examples of recent discussions: [Cappelen and Lepore, 2005], which received criticism from a lot of philosophers, which was responded to in [Cappellen, 2006a], [Cappellen, 2006b] (See there for the criticisms); [Recanati, 2007], [Travis, 2008], [Travis, 2000], [Stanley, 2000]; For overview of the discussion see [Recanati, 2004], [Bianchi, 2010].

[4]In fact, some approaches that would be called 'literalist' in the field of philosophy of language are described as contextualist in some other fields. I will however follow the terminology from philosophy of language.

[5]Cappelen and Lepore in [Cappelen and Lepore, 2005, p. 128] argue that contextualism would be inconsistent because a contextualist should hold that the sentence "contextualism is true" is false in some contexts. This argument is mistaken because contextualists do not claim that such a sentence is true in *every* context, only in those contexts in which a contextualist makes this claim.

the scientific fields of formal and computational linguistics. If contextualism turns out to be the right approach, and I believe it is, developing more formal theories for it will help its adoption in those fields of science, and in the long term it will also be relevant for applied language technology and the general ability of computers to handle human language.

1.2 Relevance within Artificial Intelligence

This research topic is very much relevant for the field of artificial intelligence. This thesis directly touches on the subjects of philosophy, linguistics, information technology and computational linguistics, and neurology, which are the four core fields within the program of Cognitive Artificial Intelligence as it is taught at the University of Utrecht. In this thesis I directly deal with discussions in the philosophy of language field and therefore also with linguistic subjects, and I have written a computer program on an artificial intelligence related topic. The other subjects are touched upon in providing supportive arguments for the main thesis.

1.3 Structure of this thesis

This thesis starts with an abstract and an introduction, which you are reading now. Chapter 2 gives an overview of the debate between literalism and contextualism and a description of some of the arguments that are used. Chapter 3 considers restrictions that a theory of linguistic meaning must satisfy. Chapter 4 gives a high level description of the proposed model and compares it with some existing models in the literature. Chapter 5 gives a detailed description of the model and ends with an example. Chapter 6 discusses how I have implemented the model in a computer program. Chapter 7 discusses limitations of the model and possible objections, and evaluates in what way the model satisfies the constraints that were layed out in chapter 3. Chapter 8 discusses ways to extend the given model and discusses further research possibilities, and chapter 9 concludes.

At the end of this thesis you will also find two appendices that contains usage instructions and other information regarding the DPSP program that was developed as part of this thesis, and that reproduce the parse rules as they are implemented by the program.

Chapter 2

Language Philosophical Background

In this chapter I will give an overview of the debate between contextualism and literalism in the philosophy of language. I will discuss the history and the main positions, and then I will discuss the most important arguments that are made.

2.1 History

The debate on the influence of context has existed in one form or another at least ever since philosophers starting to use mathematics to describe the meaning of language. The original debate, which is a predecessor of the current one, was the debate between the Ideal Language philosophers and the Ordinary Language philosophers. The Ideal Language camp was mainly concerned with formal semantics and analyzing natural language as a system of formal logic. Their main focus was on studying what the truth conditions of expressions and sentences are. This camp included people like Frege, Carnap, Tarski and Russell.[1] The Ordinary Language philosophers, on the other hand, thought that analyzing natural language as if it was a formal logic system was not the right way forward. Instead they advocated the study of language in everyday human interaction, as that would reveal what the meaning of an utterance[2] of speech was.[3] In this camp people like

[1]These pioneers were not all originally concerned with natural language, but their methods were applied to natural language by their disciples such as Montague and Davidson. ([Montague, 1970a], [Montague, 1970b], [Montague, 1973], [Montague, 1974], [Davidson, 1984])

[2]An utterance is taken to be a sentence spoken (or otherwise used) in a certain situation, so an utterance includes the contextual information from that situation.

[3]In as far as they wanted to understand language. Some were only interested in showing that certain metaphysical problems were caused by misunderstanding language.

Austin, Strawson, Grice[4], and the late Wittgenstein were to be found.[5]

Nowadays this original debate is not as prevalent anymore, as both camps recognize each others achievements. Out of the debate arose the disciplines of ‘semantics’ and ‘pragmatics’, which are now conceived of as complementary. Semantics is concerned with formal truth conditions of sentences, and pragmatics with the use of language and what the use of a sentence means or conveys in a particular context.[6]

2.2 Positions in the Current Debate

However, the debate on the influence of context continues on in a different form. The positions of the ideal language and ordinary language camp have been replaced by the positions of literalism[7] and contextualism respectively. Generally speaking, literalists hold on to the traditional distinction of semantics that deals with mostly context independent truth conditions, and pragmatics that deals with the broad contextual dependencies. Contextualists hold that such a distinction cannot be made because according to them truth conditions of sentences are pervasively context dependent. According to contextualists the carrier of truth conditions is not a sentence[8], but a speech act or utterance.[9]

Before discussing the positions of literalism and contextualism more thoroughly I will say a few words about the ‘context’ the entire debate is about. In a narrow sense, linguists sometimes use ‘context’ to talk about the linguistic context a certain word or sentence appears in, i.e. the words and sentences surrounding the linguistic object of interest in the story or dialog it is part of. ‘Context’ as relevant to the current discussion is context in its broadest form. It includes everything that can influence the meaning of language. This extremely broad view of context is in fact in itself one of the problems in the debate: if the required context that can be relevant to the interpretation of a sentence is not in any way bounded, and therefore infinite, how can we ever understand a language utterance? There would for ever be more context that would need to be processed before we could come to a conclusion regarding the meaning of the utterance. One of the

[4]Although Grice is a special case, who has said he has one foot in each camp. [Grice, 1989, “Retrospective Epilogue”, p. 372].

[5]see also [Austin, 1961].

[6]For more overview see [Partee, 2008], [Recanati, 2004, specifically the introduction], see also [Recanati, 2008].

[7]Also known as ‘semantic minimalism’, but see the rest of this chapter.

[8]Or more specifically, a disambiguated sentence with the indexicals resolved, as we will later see.

[9]Terms such as ‘literalism’ and ‘contextualism’ are used in different ways in different branches of linguistics. I will only talk about them as they are used in philosophy of language, specifically I will follow the use of [Recanati, 2004].

challenges for contextualists therefore is to limit and specify exactly what context they regard as relevant. I will come back to this question later on.

'Literalism' and 'contextualism' are two broad categories. In fact, they should be seen as the two sides on a continuum, with a lot of highly or a little bit compromising positions in between. A description of these different positions can be found for example in my [Kanis, 2011] based on [Recanati, 2004], or in [Bianchi, 2010]. I will give a short description of the most important positions below.

The most extreme form of literalism, at least theoretically, is the idea that for every context dependent sentence expressed in a given context, one can formulate a context independent eternal sentence that expresses the same thought, or in other words has the same truth conditions. This position ignores the importance of context dependence altogether. It has never been seriously defended by anyone, so it is only of theoretical interest.

2.2.1 Minimalism

The accepted form of literalism is known as semantic minimalism. This is the position that grew from combining the ideas of the ideal language philosophers and the ordinary language philosophers. This position is defended by e.g. [Borg, 2004], [Cappelen and Lepore, 2005]. According to this view, there is a class of words and grammatical constructions that is context dependent, which includes indexical words such as "I" or "now". This class consists of indexicals, ambiguity, and ellipsis, and perhaps a few other categories depending on what theory is discussed. If the referents of indexicals are fixed, and ambiguities and ellipsis are resolved, one can construct the semantic meaning of a sentence. Indexicality, ambiguity and ellipsis are understood in a strict sense, so they are taken to be finite classes of words or grammatical structures, and it is assumed to be straightforward to take what they refer to from a given context. For example the word "I" always refers to whoever utters it.

In the more formal variants of minimalism the tool of choice to represent semantic meaning are formulas of first order logic or some extension thereof. The meaning of single words is often represented as formulas in lambda calculus. In the case of lexical ambiguity one has to choose which of two (or more) formulas is the right one in a certain circumstance. These formulas are then combined (the exact way in which this combination happens being dependent on the specific theory) according to the syntactic structure of a sentence until a single formula remains that represents the meaning of the sentence. Indexicality is usually modeled by having the formula contain free variables that are not resolved by the syntactic applications. These free variables then need to be bound to an element from the context, according to the rules that apply to the indexical word in question.

The semantic representation that results is the semantic meaning of a sentence, and the formula represents the minimal conditions that must be met before the sentence can be said to be strictly literally true. Often there will be further conditions that must be satisfied before a sentence can be used in a felicitous way. These further conditions are the domain of pragmatics, where for example Gricean maxims must be taken into account. However, these further conditions can not contradict the truth conditions of the semantic representation in normal use[10].

2.2.2 Indexicalism

Indexicalists, such as Jason Stanley and Zoltan Szabò ([Stanley, 2000], [Stanley and Szabò, 2000]), acknowledge that more context dependence is needed than what minimalism can offer. They propose to solve this problem by sticking to the general architecture that minimalism proposes, but by including more indexical elements. They propose that a lot of words and constructs can have hidden indexical properties, and therefore the semantic representation can include a lot of contextual ingredients. For example, the word "tall" presumably has a hidden indexical dependency on a 'reference class', against which the tallness of the object in question should be evaluated.

Such an approach allows indexicalism to incorporate more contextual information into the semantic representation than minimalism allows, but this also forces indexicalists to specify what kinds of contextual information can be relevant and what can not. Allowing the relevant context to be infinitely large poses several problems, such as that the human mind can not deal with infinite amounts of information. Different criteria for identifying the relevant contextual information have been proposed by indexicalists,[11] but according to other philosophers they do not work.[12]

Although the indexicalist approach is theoretically less elegant due to the number of hidden variables proposed, it is nonetheless often a default approach taken by linguists who are trying to analyze a specific phenomenon and who are not necessarily aware of this discussion in the philosophy of language. When one is only looking at a single phenomenon, that may be solvable with only one or a few parameters that require contextual values, but when one tries to combine solutions for several phenomena into a single model, the number of variables can add up quickly.

[10]Where 'normal use' excludes things like irony or sarcasm. See also [Recanati, 2004, specifically sections 1.2 and 1.3].

[11] [Stanley, 2000, pp. 410–411]

[12] [Recanati, 2004, ch. 7]; see also further down in subsection 2.3.3 in this paper, where I discuss this point some more.

2.2.3 Other in-between approaches

It is also possible to define approaches that take some elements both from literalistic approaches and contextualistic approaches, for example by postulating that both the semantic representation as defended by literalism exists, and some other representation that is based on one of the contextualistic approaches. Such approaches are at risk of being vulnerable to the arguments that are used against both approaches, and of course they are also less theoretically elegant. I will not go into these positions further, but [Recanati, 2004, chapter 4] spends some more time on this.

2.2.4 Contextualism

Contextualists in general want to do away with the linguistic framework as it is proposed by literalism. Contextualists underwrite the idea of *semantic underdetermination*: that the information encoded in a sentence as such underdetermines the proposition that is expressed by an utterance of the sentence. Their attitude towards the semantic representation as defended by minimalists can differ, some contextualists argue that it is merely a useless abstraction, and others believe that such a minimalist semantic representation can not exist.

Contextualists therefore believe that there should not be a strict distinction between a semantic representation that is only mildly context dependent and pragmatic mechanisms that can not influence the core truth conditions. Exactly how they do this can differ. As an illustrating example, take the sentence "I've had breakfast". In most circumstances, what someone who utters this wants to express is "I've had breakfast *this morning*", rather than "I've had breakfast at some time in my life". According to the minimalist position, the semantic representation and thus the literal truth conditions for the sentence would correspond to the latter interpretation, because the sentence does not specify a time period. The interpretation of having had breakfast this morning is then derived pragmatically from the semantic representation. Contextualists argue that the utterer of such a sentence never intends to say anything like the minimalist semantic interpretation, and so they offer models that allow contextual information to be included in the semantic interpretation of the sentence. For a more detailed explanation and reasons for why these intuitions matter I will refer to [Recanati, 2004], especially chapter 1, and I will also touch on this in subsection 2.3.1.

One of the challenges for contextualism is to describe how context influences the semantic representation without letting context define everything, and thereby making the actual words that are used irrelevant. Recanati in [Recanati, 2004, ch. 9] describes the process of modulation, whereby a given interpretation of a phrase is adapted to the context at hand. This is not so different from what happens at the pragmatic level according to literalists, but Recanati proposes that this process also takes place during the semantic stage: every time after the representations of two words or constituents are combined according to the syntactic structure of a sentence, modulation is applied to the resulting combined representation. Then, at the next step up in the syntactic tree where more constituent combination happens, the best fitting modulated interpretations are selected for further processing. This results in a final representation of the sentence meaning that should be able to incorporate the required contextual information.

Jonathan Cohen gives a more figurative description. Based on the insulationist view as he describes it (the idea that words have a mostly fixed meaning and are not malleable) words can be seen as bricks used to build a wall. Each brick keeps exactly the same shape independent of where in a wall it is used. The interactionist view (that words can influence each other's meanings) is compared to building a wall from sand bags. Although the bag and the sand in it has restrictions on what shapes it can adopt, the final shape for each bag depends on the shape of the bags around it. If a bag is used in a different wall it will adapt a different shape based on its new sand bag environment. Similarly, the meanings of words adapt to the meanings of the words around it.[13]

Like indexicalism, contextual approaches also need to describe what kinds of context are relevant to the meaning of language. Again, different authors give different solutions. Recanati offers the availability principle: that 'what is said' (Grice's alternative for semantic representation and also used by Recanati, representing the truth conditions of a sentence) must be intuitively available to the conversation participants. Some conditions apply in that the participants must be countable as 'normal interpreters', and there are several ways to interpret 'intuitively available', which Recanati also discusses.[14]

Other scholars that fall into the category of contextualism are John Searle and Charles Travis.[15] But I will not discuss this position further at this point. I will discuss some of the arguments made more extensively in the next sections.

2.2.5 Relativism

A relatively recent development in the discussion is the development of a group of positions that claim that one cannot ascribe truth values to an utterance, but only truth conditions. This group of positions is known as 'relativism' and as 'nonindexical contextualism', and can be seen as a

[13] [Cohen, 1986, pp. 223-224], quoted in [Recanati, 2004, p. 132].

[14] [Recanati, 2004, section 1.7]

[15]e.g. [Travis, 1989], [Travis, 2000], [Travis, 2008], [Searle, 1978], [Searle, 1979], [Searle, 1980], [Searle, 1992]

variant of standard contextualism. Recanati [Recanati, 2007, p. 14] gives the following example. Take the following sentences:

> "I am French."
> "Recanati is French."

If these sentences are both uttered by Recanati, they should have the same meaning according to contextualism, since they are both uttered in the same context.[16] However, it is possible that Recanati has become delusional and has forgotten that *he* is Recanati. In that case, he could still hold the first sentence to be true, while believing that 'Recanati' is Italian because of the name. Therefore, the argument goes, although these two sentences express the same thought, that thought must still be evaluated against a background knowledge, and therefore the thought representation itself must still contain indexical elements.

A similar argument is made by MacFarlane ([MacFarlane, 2007], [MacFarlane, 2009]). The mistake, he argues, that both indexicalism and contextualism make is trying to assign truth values to utterances. Rather, what should be assigned to utterances should be truth conditions, that can be evaluated against a certain background. According to Recanati these truth conditions can contain 'essential indexicals' that are not resolved merely from the context of utterance, but depend on the circumstance of evaluation.

Relativism can be seen either as an extreme form of contextualism or as a moderate one. On the one hand relativists are willing to assign the same meaning (denoted by truth conditions) to a sentence evaluated in different circumstances of which standard contextualist would argue that they have two meanings, and on the other hand they want to use additional information in calculating the truth value of a sentence beyond only the context in which it was uttered.

2.3 Important Arguments

I will now describe the most important arguments that play a role in the discussion. The first two arguments, the 'Inappropriateness Argument' and the 'Context Shifting Argument' are the two main arguments that contextualists make against literalists. The arguments apply mostly against minimalism, and I will describe them as such. Then I will describe how

[16]For this argument to go through, we must accept that 'context' as it is used here contains only information regarding the circumstances in which the utterance was uttered. Circumstances in which the *judgement* of the truth or falsehood is made are apparently excluded. Since there are a lot of variants of contextualism there are probably some that disagrees with this assumption.

contextualists respond to indexicalism, and then I will discuss an important argument against contextualism.

It is not my goal to give a completely thorough discussion of all the arguments that are made, only to describe the ones that are important in light of my thesis.

2.3.1 Inappropriateness Argument

This argument is simple: the semantic representation according to minimalism is often not what we intuitively think we are saying, and it does not even take part in deriving it, according to our intuitions. Therefore, it is a mistake to postulate that we actually are saying such a thing. See for example the treatment of the sentence "I've had breakfast" above in subsection 2.2.4. Someone who utters that sentence will not be aware of having said something like "I've had breakfast at some time in my life", rather the interpretation of being about breakfast this morning comes unconsciously and without effort, and is usually unproblematic.[17] A theory that gives an unintuitive semantic meaning for such a sentence therefore does not match with the intuitively available data, according to contextualists.

This argument thus depends on the idea that our intuitions are relevant for a semantic representation of the kind that minimalism uses. I will not go further into the reasons of why this is so, but instead refer to Recanati who says:

> "In deciding whether a pragmatically determined aspect of utterance meaning is part of what is said, that is, in making a decision concerning what is said, we should always try to preserve our pretheoretic intuitions on the matter."[18]

2.3.2 Context Shifting

The most important argument contextualists bring to bear against literalists and specifically minimalism is called the context shifting argument. The idea is simple: Take a sentence or expression that contains no indexicals, ambiguity or ellipsis. Now find two contexts in which the sentence has a different truth value, so one in which it is true and one in which it is false or not felicitous, as judged by the intuition of native language speakers. This difference in truth value must then be caused in some way by the context, and therefore the sentence carries at least one additional contextual dependency.

[17]See [Bianchi, 2010, section 4.3]. Further discussion can be found in [Carston, 2002, ch. 5], [Recanati, 2004, p. 14], [Recanati, 2001, p. 79–80]. Searle and Travis have similar views concerning intuitive truth-conditions, cf. [Searle, 1992] and [Travis, 1997].

[18] [Recanati, 1993, p. 248]

I will give some instances of this argument as they are applied against minimalism. Their use in the debate between contextualists and indexicalists will be discussed in subsection 2.3.3.

Cat on the Mat

A lot of example sentences have been used to demonstrate this behavior. I will describe three of them. The first is by Searle ([Searle, 1978]).

Take the following sentence:

> “The cat is on the mat”

It seems like a simple sentence at first. It is true when the cat we are referring to is on the mat we are referring to. But this simplicity depends (among other things) on the background fact of us being on earth and experiencing its gravitational pull. If we were in space, even though the cat and the mat would be in exactly the same configuration relative to each other, we can question if the sentence is still true.

The sentence can again be made usable by extending the context a bit more. Assume we are in a spaceship strapped to our seat and somehow we see cats and mats floating about outside through the spaceships window. Strangely enough, they come in only exactly two configurations: one where the cat is on the mat relative to the up and down of the spaceship chair we are sitting in, and the other where the cat and the mat are in the same configuration relative to each other, but upside down from our point of view inside the spaceship chair. Now, if these cat-mat pairs are drifting by at regular intervals, and you asked “which attitude is it now?”, it would be felicitous to answer with “the cat is on the mat”.

The sentence contains no indexical words, and we can assume there are no ambiguities either. This is a problem for minimalists because according to them, such a sentence has only one semantic representation and it is the same in all situations. If for this semantic representation only the cat and the mat in question are important, and perhaps something like the presence of a world or a gravitational field, this sentence should have the same truth value in the case where cat-mat pairs are floating about in space in isolation and the described situation in which we are watching the cat-mat pairs from our spaceship.

Indexicalists, in turn, answer to this challenge by positing that a construction such as “is on” has a hidden indexical variable that contains a reference to the thing relative to which the ‘on’-ness is to be interpreted, or more precisely what background we use to determine what is up and down. In the normal case of a cat and a mat here on earth, this background would be the earth itself. If the cat-mat pairs were floating about in space in isolation there would be no felicitous way to resolve the indexicality, but there would be again in the case where we are watching from our spaceship.

Cut the Grass

A second example also comes from Searle, and has also been widely reused by many others. In the two sentences

> "John cut the cake"
> "John cut the grass"

the two uses of "cut" have a different meaning. In the first it means something related to cutting a thing into pieces with a knife, in the second sentence we are talking about mowing the lawn. But the small piece of contextual information provided by the words "cake" and "grass" is not enough to fix the meaning of "cut". Take the following situation:

> Suppose you and I run a sod farm where we sell strips of grass turf to people who want a lawn in a hurry. Suppose I say to you: "Cut half an acre of grass for this customer". I might mean not that you should *mow* it, but that you should slice it into strips as you could cut a cake or a loaf of bread[19].

In the context set up here, "cut" again has a meaning that is very close to that in "cut the cake", rather than to "mowing the lawn".

As in the previous example, the sentence contains no indexicals or other constructions that according to minimalism should be context dependent. Therefore the sentence of "John cut the grass" should only have one semantic representation, and therefore should have the same truth conditions in the case where it is used to talk about cutting the lawn and in the case of the grass sod farm. However if, in the context of the sod farm where a request was made to you to cut a grass sod to size, and what you actually did was mow the piece of grass, you would not have fulfilled the request. And in the same way, if you were asked to "cut the grass" in the lawn mowing sense and you actually cut some pieces of grass sod out of the lawn, you would not have fulfilled the request.

Indexicalists respond to this in the same way as with the previous example. They propose that the semantic representation of 'cut' contains a hidden indexical, that refers to the 'mode of cutting' that is to be considered. This mode of cutting then differs for the normal case of mowing the lawn and when cutting grass sod.

Milk in the Fridge

A third example of such a context shifting argument comes from Charles Travis in [Travis, 1989, p. 18, 19] (also treated in [Cappelen and Lepore, 2005]). Imagine the fridge in the house has only a small puddle of milk on its floor, but no other milk in it. Now take the following two contexts:

[19] [Searle, 1980, pp. 224–225], quoted in [Recanati, 2004, p. 133]

- Hugo is dejectedly stirring a cup of black coffee. Noticing this Odile says *"There is milk in the fridge"*.
- Hugo has been cleaning the fridge. Odile opens the fridge door and says *"There is milk in the fridge"*.

Although what was spoken was the same on both occasions, intuitively Odile has spoken truth in the second context, but not in the first. The word "milk" must therefore refer to something like "milk in a bottle usable for coffee" in the first context, but not in the second, where it refers to any kind of milk, or maybe "milk not properly located in a container".

The argument applies against minimalism in the same way as the two earlier ones: "There is milk in the fridge" contains no indexicals or ambiguities, and so should have only one semantic representation and one set of truth conditions. Yet in the two situations in the example, in one Odile can be said to have spoken truthful but not in the other. Indexicalists also respond similarly, by proposing a hidden indexical for the kind of milk that is to be considered.

2.3.3 Arguments against Indexicalism

The two arguments above do not apply as cleanly against indexicalism as against minimalism. For every concrete example, indexicalism can posit that there is a hidden indexical dependency and thus come up with a semantic representation that conforms to our intuitions.

A contextualist can respond to this move by constructing a different context shifting situation that is not covered by the indexicalist's definition of what words and constructs contain hidden indexicals. The indexicalist in turn can extend his theory to cover these new cases. But the indexicalist has a big disadvantage in this game, because he has to make sure his theory covers *all* possible cases, while a contextualist only has to find *one* exception.

An indexicalist cannot postulate an infinite number of hidden indexicals that work in the way that minimalism proposes, because each indexical requires a description of how it influences the semantic representation of a sentence. An infinite number of such indexicals would therefore imply an infinitely large theory, and that would prohibit any systematic theorizing about the meaning of language.

What is therefore needed are general rules that can apply to whole classes of perceived context dependence at once. The first task indexicalism therefore has is deciding which kinds of context dependence are instances of hidden indexicals or other minimalist mechanisms, and which ones are caused by pragmatic effects. The kinds of context dependence that are caused by pragmatic effects should then not have an influence on the underlying semantic representation and thus on the sentence's truth conditions. According to Recanati, the proposals for such general rules that have been made do not hold up on closer inspection.[20] As this is an argument that is internal to the workings of indexicalism and is not directly related to contextualism, I will not discuss it here but refer to the literature.

2.3.4 Formal Description of Contextualist Models

After having described the main arguments made by contextualists, I will now turn to discuss some of the arguments made by literalists against them.

Literalism originated out of the tradition of ideal language philosophy, which was originally based not on attempting to understand natural language, but on understanding different kinds of logic. The logic tools developed there were then applied to natural language, most famously by Montague and those following in his tradition.[21] As a result formalizing the theories that were developed has always been an important branch of the research that is being done in this tradition.

For contextualism this is not so much so. While for example the Gricean maxims that came out of the ordinary language philosophy program can be very useful, to formalize them would require some way of formalizing a large enough part of the context and the dialog that is taking place, and having a way do decide which parts of contextual information are relevant in what cases. A general problem that all context related theories suffer from has always been that there is just too much contextual information that could be relevant, and that it is impossible to describe it all in mathematical terms.

If a theory of meaning is to give a *full* description of the 'meaning' (for some sense of 'meaning') of natural language, it will ultimately need to be computable. A theory of meaning should be able to tell us what a certain sentence (possibly used in a certain context) means. The only way to do that without relying on the subjective judgment of a human interpreter is by using math.

Such a complete, and as of yet hypothetical, theory of language should also be able to explain the intuitive judgments people make regarding the meaning of language, otherwise it is nothing but an arbitrary artificial language that happens to look like a natural language, but that does not in fact have the same meaning. Or in other words, a *Luftgebäude*[22].

[20]In [Recanati, 2004, section 6.2], Recanati makes an argument similar to what I have made above. In chapter 7 of idem Recanati attacks general criteria for deciding what kinds of context dependence are caused by hidden indexicals, as defended by Stanley and Szabó ([Stanley and Szabò, 2000]) who are the most important defenders of indexicalism.

[21]see [Montague, 1970a], [Montague, 1970b], [Montague, 1973], [Montague, 1974].

[22]Kripke writes the following: "I find myself torn between two conflicting feelings—a 'Chomskyan' feeling that deep regularities in natural language must be discoverable by an appropriate combination of formal, empirical, and intuitive techniques, and a contrary (late) 'Wittgensteinian' feeling that many of the 'deep structures', 'logical forms', 'underlying semantics' and 'ontological commitments', etc., which philosophers have claimed to discover by such techniques are *Luftgebäude*." ([Kripke, 1976, 412 n. 56], quoted

The debate between literalism and contextualism takes place mainly in the field of philosophy of language. As such the arguments made are mostly of philosophical origin, and the proposals for contextualist models are usually descriptions at a philosophical level. While defenders of contextualism do try to give rigid descriptions for their models they are still much vaguer than what researchers in more formal branches of linguistics use.

There are also more sceptically inclined people who would argue that a model that properly describes the meaning of natural language will necessarily also be about as complex as the human brain. Obviously I disagree with them and I think that while a model that gives exactly the same results as the human brain may need to be equally complex, it is possible to define relatively simple and manageable models that can answer a lot of the important questions.[23]

in [Pietrosky, 2005, p. 239]).

[23]For example see [Madsen, 2009]. Although he is talking about machine translation, the same arguments also apply to formalizing the meaning of language

Chapter 3

Constraints on a Formal Model of Language

The central goal of this thesis can be found in the argument described previously that there are no good ways to formalize contextualist approaches. My goal with this thesis is to attack this argument head-on, and give a description of a contextualist model that can be implemented in a computer, while not jeopardizing the other pro-contextualism arguments. I will also provide such a computer implementation as a proof-of-concept.

In this chapter I will consider some weaker and stronger constraints for both a theoretical model of the meaning of language and for a program that implements such a model. I will start by laying out a specific non-requirement, and also discuss some soft constraints that should be seen more as sources of inspiration, and then move on to stricter constraints.

3.1 Non-Requirements

Since one of the things I have done is to build a computer program, I will specifically look at some issues related to the computational side.

There have been lots of attempts to implement natural language semantic systems in limited domains. Some examples are SHRDLU ([Winograd, 1971], [Charniak, 1972]), and a more recent one is Dipper ([Bos et al., 2003]). Both of these are based on literalistic traditions, in the case of Dipper the system makes use of Discourse Representation Theory[1] and proof systems that try to rule out inconsistent or uninformative interpretations of sentences. SHRDLU is based on the formal linguistic ideas from the 70s. Both of these systems are limited by a small knowledge domain and both can only handle a relatively strict subset of English.

The program I have implemented does not attempt or succeed in im-

[1] [Geurts and Beaver, 2011], [Kamp, 1981], [Heim, 1982]

proving on these systems on these practical measures, it only tries to do something superficially similar but based on a different theoretical footing.

3.2 Statistical Linguistic Systems

Besides linguistic systems that come out of academia as research tools there is also a lot of generally usable applied language technology in existence. Examples are speech recognizer systems or automatic translation systems such as Google Translate[2]. What these systems have in common is that they do not actually deal with semantics or meaning of the sort that linguistics is traditionally interested in. What they do is transform one form of language into another form of language. In the case of speech recognition this is from sounds into text, and in the case of translation from text in one language to text in another language. How that transformation is made is determined by a statistical model that calculates the most likely target representation. The reason why they do not do 'real' semantics is simple: it works better without it. A well known quote on this is the one by Frederick Jelinek, who worked on language technology systems for IBM: "Every time I fire a linguist, the performance of the speech recognizer goes up"[3]. Google engineers have also tried different linguistic approaches without much succes.[4]

The success of these statistic methods in language processing, and other general considerations regarding information theory, have led some to propose that the brain can be seen as a statistical inference engine.[5]

The statistical linguistics approach in general is very different from the approaches taken by traditional linguistics. To bridge this gap, there have been attempts at developing statistical techniques that include a semantic

[2]http://translate.google.com (accessed Aug. 2013) For an academic publication on which it is based see e.g. [Och et al., 1999].

[3]From http://en.wikipedia.org/wiki/Frederick_Jelinek. (accessed Aug. 2013) The exact context of the quote is unclear, but wikipedia notes "Although its fame and iconic status are undisputed (it was for example used as the title of a 1998 speech by Julia Hirschberg),[1] its context is unknown and its specific wording and dating are unclear. According to Daniel Jurafsky and James H. Martin, Jelinek himself recalled the quote as "Anytime a linguist leaves the group the recognition rate goes up" and dated it to December 1988 (Wayne, Pennsylvania), further noting that the quote did not appear in the published proceeding,[2][3] whereas Roger K. Moore gave the wording as "Every time we fire a phonetician/linguist, the performance of our system goes up" and dated it to an IEEE Automatic Speech Recognition and Understanding workshop held in 1985.[4] According to Steve Young, "the story goes that one day one of his linguists resigned, and Fred decided to replace him not by another linguist but by an engineer. A little while later, Fred noticed that the performance of his system improved significantly. So he encouraged another linguist to find alternative employment, and sure enough performance improved again. The rest as they say is history."[5]" (see wikipedia for the applicable references).

[4]Unfortunately I am unable to find the reference, but I remember seeing it in a Google TechTalk video online. See http://www.youtube.com/user/GoogleTechTalks (accessed Aug. 2013).

[5]See e.g. [Doya et al., 2007].

representation. One such approach is to use as the semantic representation of a word of interest, the bag of words that surround it.[6] This results in a semantic representation that can be used for some things—for example the different meanings of the word “bank” would show a very different set of surrounding words, with one containing words related to finance and the other to furniture—but it is still a very much impoverished representation compared to logical formulas. Making complicated derivations based on bag-of-words representations is a whole lot harder than doing the same with logical formulas.

While using statistical approaches is not a strict requirement, the success of this approach argues in favor of using it for the kinds of problems it is good at. For my project of developing a formalizable model of contextualism I have taken this information as an additional guide, and I have come up with a model that allows, at least in principle, for the use of statistical/bayesian techniques, but that still has a rich representation of meaning that can be used for example for making logical derivations.

3.3 Neurological Realism

A stricter constraint that any theory of language must meet is that it must be possible to implement such a theory in architectures similar to the human brain, and provide performance comparable to the actual performance humans have on linguistic tasks. Of course a theory that does not meet this criterion can still be useful from a theoretical perspective, but it cannot be the correct theory of what actually happens in someones head.

What the human brain is and is not capable of is not easy to say in detail, but one thing that is known is that it is a massively parallel system with very slow sequential performance, at least compared to contemporary computers. Most brain neurons can fire at a rate of a few hundred hertz,[7] so that is orders of magnitudes slower than current computer chips that run at speeds measured in gigahertz. Since humans are capable of acting on a linguistic stimulus within a few hundred microseconds, that only leaves a few dozen linear steps at most that the brain has to parse and interpret the language input. On the other hand, the brain is massively parallel, with billions of neurons and trillions of synapses operating in parallel.

The implication of this for any theory on natural language processing that tries to describe what humans actually do, is that the theory must be implementable on a massively parallel system.

[6] [Manning and Schütze, 1999, ch. 14]
[7] [Coon, 1989]

3.4 Evolutionary Realism

Any model that purports to model a system in the body or brain has to adhere to another restriction: the system has to have evolved somehow, so the model must allow for an evolutionary origin. What exactly can have evolved and what cannot is hard to say for something so general as the human linguistic capability, so I will limit myself to the most general evolutionary constraints.

Biological changes start with mutations. The most likely mutations are small mutations[8]. These small mutations can over time accumulate to produce larger collective changes. But for natural selection to promote a specific mutation to a higher frequency within a population, that specific mutation by itself must provide an increase in the probability that an individual will successfully procreate (also known as 'fitness'). What does not happen is that one mutation rises in frequency in a population because it will later on be beneficial when it is combined with a second mutation that does not yet exist. Evolution is not directed at a goal and evolution does not plan ahead. This means that any complex system that has evolved by accumulating small mutations must have shown a gradual increase in utility, the more of the mutations accumulated.

Larger mutations can also happen, for example pieces of DNA can 'jump' from one location in the DNA string to a different location, which can give them a new function in a different protein. Sometimes small mutations in a regulatory area in the genome can have very large effects, such as disabling an entire gene or cutting the protein the gene codes for in half. Other large changes can happen through viruses, which can transplant DNA matter from one organism into a different unrelated organism.

All these mechanisms have one thing in common: they take existing pieces of DNA with existing functionality and allow it to be used for a different function. However, the pieces must first have come into existence for their original functional purpose. In general it can therefore be said that evolution does not allow complex systems to arise out of nothing. Simpler, yet functional, systems must have preceded it, or parts must have had a different function in some other system before they were re-used in the system in question.

Applied to the brain, these restrictions apply to those systems and structures that are coded for in our DNA, so the ones which we are born with. Knowledge that we learn has not evolved, so it does not need to abide by this restriction, however the capacity to learn things does.

In the case of linguistics, then, this restriction applies to the elements of a system that are postulated to be innate, and it applies especially to those

[8]Or more technically single nucleotide mutations, mutations which change, insert, or delete a single letter from the DNA code.

elements that are supposed to be unique to humans. Systems and parts of systems that are shared with different species probably also existed in our shared ancestors, so they have had a very long time to evolve. Systems that are unique to humans can only have started to evolve once the human lineage split off from our nearest ape relatives, and so have had relatively little time.

After explaining the linguistic model I have developed, I will also argue that it is realistic from an evolutionary point of view.[9]

3.5 Learnability

As a last constraint that I will discuss, a linguistic model must be learnable in as far as it is not innate. Children are able to learn language relatively quickly with a limited amount of input. The input usually consists of positive uses, and not nearly as much of negative examples or corrections. Despite this, children learn language without apparent difficulty.[10]

A lot of studies have been done regarding how children learn language and what kinds of language they can and cannot learn. There is a lot more to say on this constraint than what I will discuss in this thesis due to time limitations.

[9]Arguments of a similar type have been used before in the literature to question linguistic models, for example [Parker, March 2006], who argues that Chomskyan syntactic minimalism does not fit evolutionary restrictions.

[10]For some of the literature on this subject, see [Bertolo, 2001], [Fletcher and MacWhinney, 1995] and [Ritchie and Bhatia, 1999].

Chapter 4

Overview of the Model

A core principle behind the model I have developed is that the human mind contains a conceptual representation of the world around it. This conceptual structure is not specific for language processing, but is the general purpose representation that minds use to reason about the world around them. It is stipulated to contain at least all declarative knowledge that someone has. A meaning of a sentence, then, is some kind of datastructure that is part of or is embedded in this conceptual representation. Parsing a natural language sentence or expression becomes equivalent to finding the right mapping from the input to the conceptual structure.

In this chapter I will start with a broad overview of the model and compare it to some related proposals in the literature. The next chapter contains a much more detailed description of the model.

4.1 Conceptual Graph

For my conceptual structure I have chosen a graph representation. Nodes in the graph are concepts, and there can be all kinds of relations between them.[1] The concepts represent (among other things) objects in the environment, and in that way the conceptual graph models the external world.

The choice for the graph I have made is in part arbitrary. I have chosen this representation mainly because it is simple and it works well enough, but I do not defend the position that the graph structure as represented here is a highly accurate model of how the brain represents information. What we know about the brain in fact makes it likely that different kinds of representations are used in different parts of the brain, for instance there are a lot of differences in the behavior of the short term working memory and long term memory, so it seems likely that they also use different representations for the information they deal with. What I will defend is that the brain has

[1]These relations themselves also correspond to concepts. The exact structure I use will become more clear when we come to the actual implementation.

within it an encoded representation of the world around it, that thought processes operate on this representation, and that ‘meanings’ of sentences should be seen as structures within this representation.

The brain is finite, and therefore any kind of structure within it also has to be finite. I therefore take the concept graph to be finite as well, encoding a finite amount of information. However I propose that the graph can be extended when needed, for example in response to linguistic input, but also in response to sense data or just due to thought processes. For example we know there are infinitely many numbers, but in this model only a finite number of them can be represented as individual concepts at any one time. But we can think of new numbers as we please, so such thoughts cause new concepts to be inserted into the graph as needed.

4.2 Parsing

The parser is a relatively thin layer that tries to map input words and sentences onto concepts in a recursive fashion. The choice of which concepts is in part determined by which concepts exist in the graph and what relations they have. This means that the parser automatically takes contextual information into account as that contextual and situational information is also represented in the concept graph. All information that is present in the concept graph is in principle available to the parser.[2] This construction also provides a natural limitation on what contextual information is relevant: only information that is represented in the conceptual graph (i.e. the brain) can influence the interpretation.

Since parsing is a task of mapping one set onto another set, it is (at least in principle) possible to apply probabilistic techniques in the model to calculate the most likely mapping rather than rule based techniques. This and the next two chapters give some examples of how this could work, but other ways may be possible as well.

4.3 Name

I have decided to name the general framework “direct probabilistic semantics” or DPS. Calling it ‘the model’ all the time in this thesis did not seem like a good idea, so naming it is better. The name emphasizes the direct interpretation, without going through several intermediate logical or other forms and which requires the existence of some kind of world representation in the mind, and the possibility of using probabilistic techniques in making that interpretation.

[2]That is in contrast to e.g. [Asher and Pustejovsky, 2005, p. 3], who argue for a ‘glue logic’ that specifically does not have access to all general purpose semantic knowledge.

4.4 Related Proposals in the Literature

The proposal as described here is not is not totally new in the literature. Proposals have been made before that have quite a bit of similarities with what I propose here. However as far as I am aware none of them have been developed formally anywhere.

One of them is the 'conduit metaphor', a metaphor described by Michael J. Reddy [Reddy, 1979][3]. He argues that people tend to conceive of language as a conduit, through which thoughts can be transferred to other people. According to Reddy such a conception is naive. Thoughts cannot be packaged into language and transmitted to another person.

A better way to look at language, Reddy argues, is the 'toolmaker metaphor'. In this metaphor several people who live in very different environments and who do not have a shared language can only communicate by sending drawings to each other. They use this to send each other blueprints of tools they have built, but since each lives in a different environment where different materials are available and different problems are important, they do not always interpret the blue prints as intended by the sender. Each applies and adapts the blueprints he receives to the problems in their environment and the materials that are available. The people then also send their adapted blueprints back to the others, and the process repeats. The tools each person ends up with are partially similar to each other, but also different because they exist in different environments.

If language worked like a conduit, the people in the metaphor would be able to send replicas of the tools themselves to each other. Each would then be able to see exactly what the tool looks like and what materials it is made of. But since they are limited to sending blueprints, their communication is more difficult, yet in no way impossible.

A second similar proposal is expressed in [Pietrosky, 2005], where Pietrosky argues against a too close relation between meaning and truth and in favor of linguistic meaning being something that is internal to the interpreter and subject to a massive interaction effect between properties of the utterance in question and other factors not part of the utterance. Pietrosky argues within a line of reasoning known as 'internalist semantics', a view that is also argued for by Chomsky in [Chomsky, 2000]. However this idea apparently also suffers from the vagueness of being a philosophical theory that started out from disagreeing with other semantic approaches and not being very well formalized.[4] Although my approach does not match exactly with this form of internalist semantics, I think it is close enough to be helpful

[3]See also the wikipedia article on the conduit metaphor: `http://en.wikipedia.org/wiki/Conduit_metaphor`.

[4]See also the description in the Stanford Encyclopedia of Philosophy ([Speaks, 2011]): `http://plato.stanford.edu/archives/sum2011/entries/meaning/#ChoIntSem`, and [Pietrosky, 2003].

in clearing up some of the vaguenesses involved.

4.5 Core Differences with Standard Approaches

One of the central differences between the above approaches and the DPS model on the one hand, and most mainline approaches to semantics on the other hand, is that the “meaning” of language is explicitly described as something personal rather than a mathematical entity that exists independent of any subject. A second difference is that (at least in the DPS model) logical formulas are not used as denotations of meanings, as is common in many approaches. This implies that the truth conditions of sentences take a much less central role. For the DPS model, in the case of some sentences such as predicative sentences, it can be possible to evaluate the conceptual structures into which these sentences are interpreted to get a truth value, but this value does not represent some kind of absolute truth value, but rather whether the sentence is *believed* to be true by the listener. The actual truth of a sentence depends on how accurately the conceptual graph models the external world, which is something that cannot be assessed easily (see for a more detailed discussion of this point subsection 7.1.2 on page 64).

The DPS model also differs from many existing approaches in that there is no direct way of defining what the ‘correct’ interpretation of a sentence is. Communication can be said to have succeeded to the degree in which the meaning the listener ascribes to an expression is isomorphic to the meaning that the speaker tried to convey. One can say that the correct interpretation of an expression is whatever the speaker meant, but there is no primitive notion of what the correct interpretation for a string of words is according to the English language, or even of what a language is.

A ‘language’ can only be defined in a statistical sense as the group of people who all interpret certain word combinations in isomorphic ways, and the ‘correct’ interpretation for a sentence as used in a specific context then is what this group says is the correct interpretation. With a language such as English or other languages that are spoken by large groups of people, there will be an overwhelmingly large majority that agrees on a specific interpretation for most non-complicated sentences. But there will also always exist example sentences, likely more complicated or contrived ones, on which there is no large majority consensus. It is not a failure of linguistics that it fails to uncover the ‘correct’ interpretation of such sentences, it is just that the statistical nature of what a ‘language’ is means that for some sentences there is simply no fact of the matter to discover regarding its ‘correct’ English interpretation.[5]

[5]Chomsky in [Chomsky, 1986] describes a similar position, but in respect to syntax rather than semantics.

Chapter 5

Detailed Model Description

In this chapter I will describe the details of the DPS model. At the end I will discuss an example to illustrate how the proposed mechanisms work.

5.1 Concept Graph

The conceptual graph needs to be able to represent any kind of information that is represented in the human brain. Therefore it needs to be very general. The graph structure I have chosen consists of a set of nodes and a number of edges over them. The nodes in the graph correspond to concepts, and as such they represent (among other things) objects in the environment, but also more abstract things such as object properties and relations between objects.

Since there can be different kinds of relations between objects, the graph needs to be able to represent different kinds of relations. Since not all relations that need to be represented will be symmetric, we will use a directed graph. The edges of the graph will be labeled with sets of concepts themselves to differentiate different relations.

Figure 5.1 gives a graph description of a simple scene with three objects in it, a green triangle, a red triangle, and a red square. Note that the names in the node are only there to identify the nodes to the reader more easily, they do not form part of the graph structure.

The graph as displayed here only allows for the representation of a single state of the world (at least if the world is represented as in this example). To allow for a straightforward description of more than one state of the world, or more than one world, we can extend the labels on the edges to be sets of tuples of concepts, where the first element of the tuple is the world, and the second is the relation. Worlds can be encoded as concepts in the same way as relations are. So, in Figure 5.2, we see a graph that contains two worlds, ‘t1’ and ‘t2’. If ‘t1’ and ‘t2’ are interpreted as representing two points in time, this graph represents a scene in which at t1 there are a green triangle,

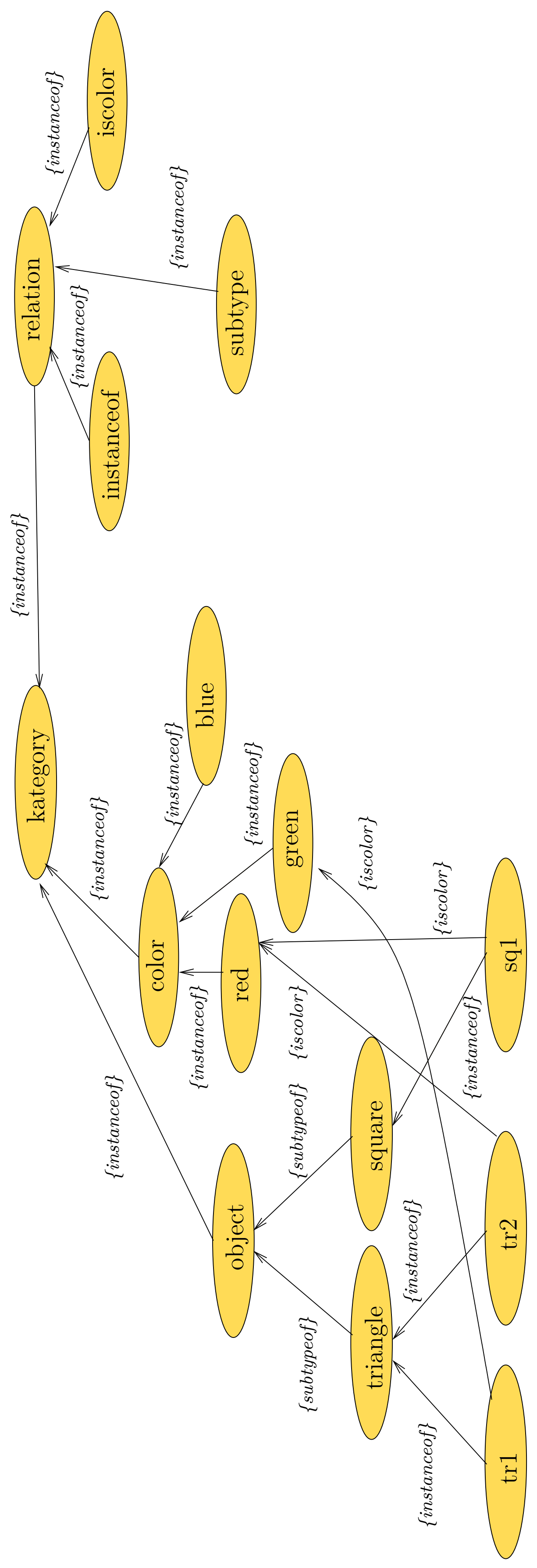


Figure 5.1: Concept Graph

Figure 5.2: Concept Graph (multiple worlds)

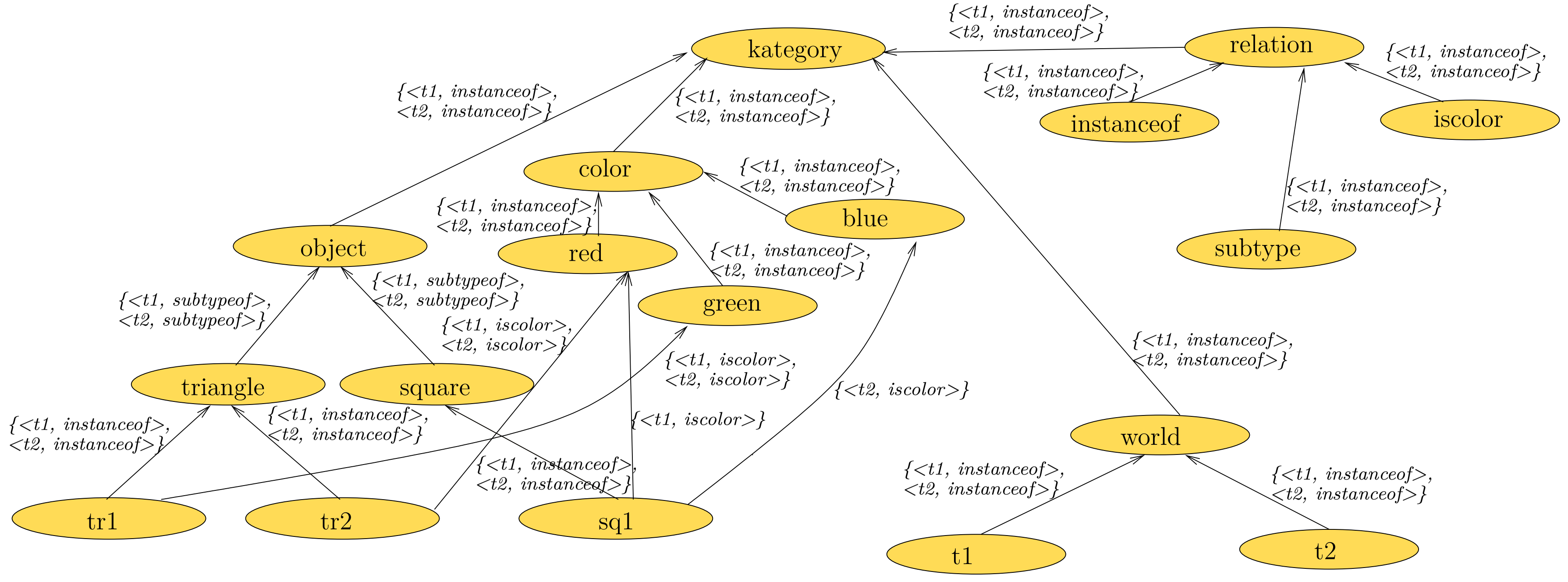
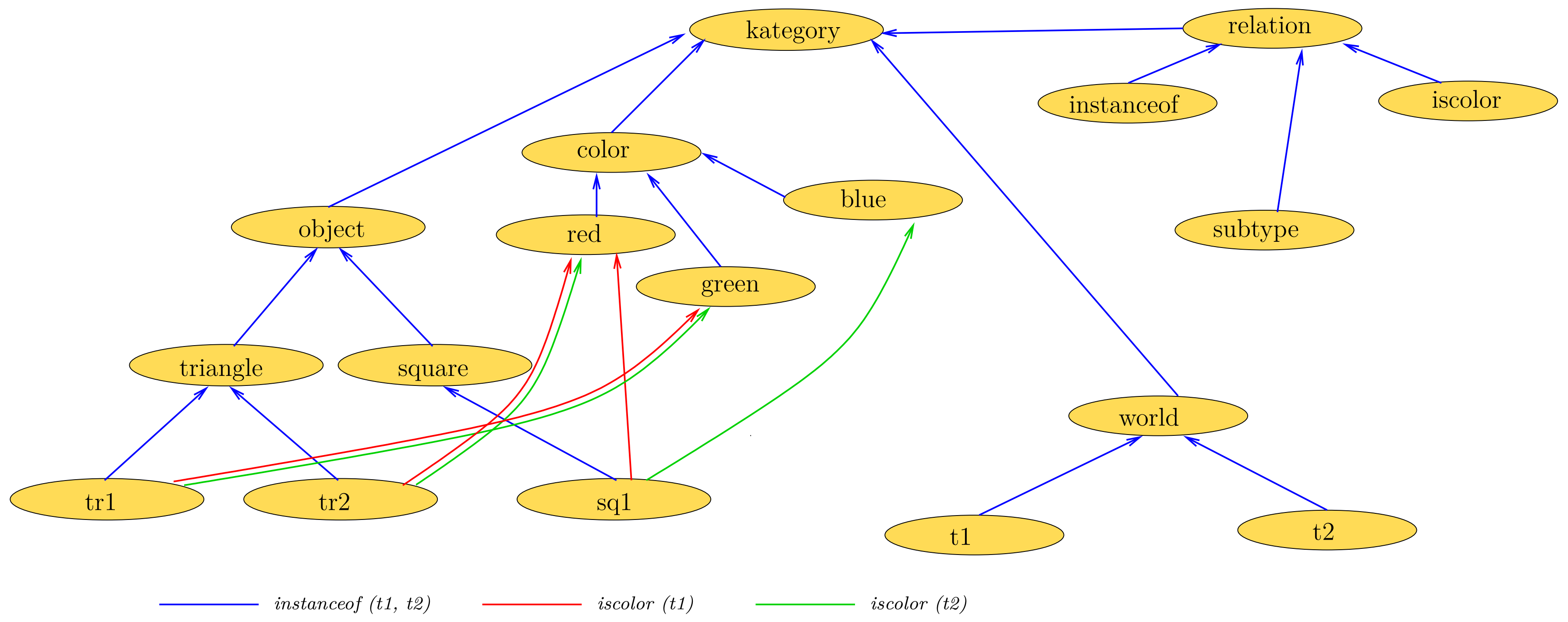


Figure 5.3: Concept Graph (separate relations interpretation)



a red triangle, and a red square, and at t2 the red square has become blue.[1]

Mathematically, it is possible to use a simpler representation than the graph with labeled edges as described above. We can represent the graph as a set of nodes plus a set of relations. The relations can be encoded as 4-tuples

$$\langle \mathit{world}, \mathit{type}, \mathit{subject}, \mathit{target} \rangle$$

where each of *world*, *type*, *subject*, and *target* are concepts. *World* and *type* represent the labels that are connected to an edge, and *subject* and *target* represent the nodes that the edge connects. The graph from Figure 5.2 can thus be represented as the following mathematical object

$$\langle C, R \rangle$$

where

$$R \subseteq \{\langle w, t, s, g \rangle \mid w, t, s, g \in C\}$$

The set C consists of 18 items, which for the ease of understanding the graph can be identified with the following names:

kategory[2]	triangle	iscolor
object	red	tr1
color	green	tr2
relation	blue	sq1
world	subtypeof	t1
square	instanceof	t2

The set of 4-tuples R is the following:

$$\begin{array}{ll}
\langle \text{t1}, \text{instanceof}, \text{kategory}, \text{kategory} \rangle & \langle \text{t1}, \text{instanceof}, \text{blue}, \text{color} \rangle \\
\langle \text{t1}, \text{instanceof}, \text{object}, \text{kategory} \rangle & \\
\langle \text{t1}, \text{instanceof}, \text{color}, \text{kategory} \rangle & \langle \text{t1}, \text{instanceof}, \text{iscolor}, \text{relation} \rangle \\
\langle \text{t1}, \text{instanceof}, \text{relation}, \text{kategory} \rangle & \langle \text{t1}, \text{instanceof}, \text{subtypeof}, \text{relation} \rangle \\
\langle \text{t1}, \text{instanceof}, \text{world}, \text{kategory} \rangle & \langle \text{t1}, \text{instanceof}, \text{instanceof}, \text{relation} \rangle \\
 & \\
\langle \text{t1}, \text{instanceof}, \text{square}, \text{object} \rangle & \langle \text{t1}, \text{instanceof}, \text{t1}, \text{world} \rangle \\
\langle \text{t1}, \text{instanceof}, \text{triangle}, \text{object} \rangle & \langle \text{t1}, \text{instanceof}, \text{t2}, \text{world} \rangle \\
 & \\
\langle \text{t1}, \text{instanceof}, \text{red}, \text{color} \rangle & \langle \text{t1}, \text{instanceof}, \text{tr1}, \text{triangle} \rangle \\
\langle \text{t1}, \text{instanceof}, \text{green}, \text{color} \rangle & \langle \text{t1}, \text{instanceof}, \text{tr2}, \text{triangle} \rangle
\end{array}$$

[1]So, I have chosen to make relations world-relative but concepts themselves absolute. An intuitive justification for this is that we tend to think of objects as keeping the same identity over time, rather than each instant in time consisting of different but related objects. If an object does not exist in every world that is modeled in the graph, this can be modeled by having an 'exists-in' relation between objects and worlds where appropriate.

[2]The name 'kategory' is a pun on 'Kantian category'.

$\langle \mathrm{t1}, \mathrm{instanceof}, \mathrm{sq1}, \mathrm{square} \rangle$

$\langle \mathrm{t1}, \mathrm{iscolor}, \mathrm{tr1}, \mathrm{green} \rangle$
$\langle \mathrm{t1}, \mathrm{iscolor}, \mathrm{tr2}, \mathrm{red} \rangle$
$\langle \mathrm{t1}, \mathrm{iscolor}, \mathrm{sq1}, \mathrm{red} \rangle$

$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{kategory}, \mathrm{kategory} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{object}, \mathrm{kategory} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{color}, \mathrm{kategory} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{relation}, \mathrm{kategory} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{world}, \mathrm{kategory} \rangle$

$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{square}, \mathrm{object} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{triangle}, \mathrm{object} \rangle$

$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{red}, \mathrm{color} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{green}, \mathrm{color} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{blue}, \mathrm{color} \rangle$

$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{iscolor}, \mathrm{relation} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{subtypeof}, \mathrm{relation} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{instanceof}, \mathrm{relation} \rangle$

$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{t1}, \mathrm{world} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{t2}, \mathrm{world} \rangle$

$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{tr1}, \mathrm{triangle} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{tr2}, \mathrm{triangle} \rangle$
$\langle \mathrm{t2}, \mathrm{instanceof}, \mathrm{sq1}, \mathrm{square} \rangle$

$\langle \mathrm{t2}, \mathrm{iscolor}, \mathrm{tr1}, \mathrm{green} \rangle$
$\langle \mathrm{t2}, \mathrm{iscolor}, \mathrm{tr2}, \mathrm{red} \rangle$
$\langle \mathrm{t2}, \mathrm{iscolor}, \mathrm{sq1}, \mathrm{blue} \rangle$

There are a few things to note about this representation. First of all, concepts (the nodes in the graph) can represent very different kinds of things. What a concept represents depends only on the labels of the relations it has. There are no further types associated with the concepts apart from those encoded in their relations. This makes the graph very flexible and allows it to represent many different things.[3]

The graph can be interpreted as a graph with labeled edges, but a different and for our purposes simpler interpretation is to see the graph as a set of nodes on which multiple relations are defined. A relation in this interpretation consists of all the edges that have a specific tuple in their label sets. Thus, we can talk about the 'iscolor' relation in world 't1', or the 'instanceof' relation in world 't2'. Another natural way to talk about a relation that exists in multiple worlds that represent different points in time is to talk about for example the 'iscolor' relation that changes in time, and is different at time 't1' and at time 't2'. This interpretation is displayed in Figure 5.3.

From here on I will talk about the graph and the set of tuples R as containing multiple relations, for example I will talk about 'sq1' as having a 'iscolor' relation to 'blue' at time 't2'. The 'instanceof' relation functions as a description of the types of concepts. As such I will talk about 'sq1' as being a 'square'.

[3]If the relations are given the interpretation as suggested by their names in Figure 5.2, the flexibility of the graph also allows representation of states of the world that cannot exist, such as a thing being a square triangle. I consider this a feature, since human thought is not always consistent. However I will not explore this possibility in depth and I will here only consider "sensible" graphs.

The graph as represented here has an implicit restriction that any concepts that appear as the first element of an edge label tuple must be a world, i.e. must have a relation with an 'instanceof' type to the 'world' concept, and similarly a concept that appears as the second element of such a label must be a relation. In principle it would be possible to define graphs similar to the ones here that violate such conditions, but I will not consider them here. Such graphs should be seen as not well formed for the purposes I have with them here.

Similarly this representation allows for things like the 'instanceof' relation changing from 't1' to 't2' so that e.g. 'color' stops being a 'kategory'. I will also not consider such changes. All kategories and 'instanceof' relations should be considered to be the same in every world, or, in other words, immutable.

As the listing of the relation tuples above shows, having multiple worlds can quickly make for a large number of relation tuples. All but one of the relation tuples in R above are the same for 't1' and 't2', so clearly this is not a very efficient representation if there are a large number of worlds. While this is not a concern for math, any kind of implementation would need to be more efficient in storing such a graph.

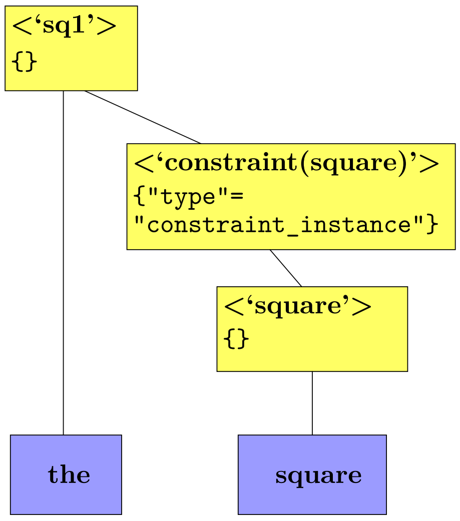
As a last note, one may wonder how this graph structure can represent the external world if the nodes themselves do not carry any name or meaning apart from their relations, or if they are not grounded in any way. To model the external world, some of the nodes should be treated specially, such as 'kategory' and instances thereof, and things like colors. It is assumed that instances of 'kategory' have a special status as things that are innately built into our brain. For sense experiences specific concepts such as colors are assumed to be innately connected to the sense organs, thus grounding the model in our sense experiences. The model as presented here does not implement this in order to make things no more complex than needed.

5.2 Parser

As noted in section 4.2, the parser has the task of mapping input sentences into concepts. The parser algorithm is based on a standard chart parser. The input to the parser is a list of words. The parser consists of a set of pattern matchers. Each matcher looks for a specific pattern in the parse tree that is being constructed, and if it finds one it creates one or more new parse tree nodes. (Multiple parse tree nodes can be used to represent ambiguities.) As such, such pattern matchers corresponds to a production rules in standard parsing algorithms.

Contrary to most natural language processing systems, a parse tree is not a self-contained mathematical entity, rather it contains concepts from the concept graph. Each node of a parse tree consists of a number of items,

Figure 5.4: Parse Example (“the square”)



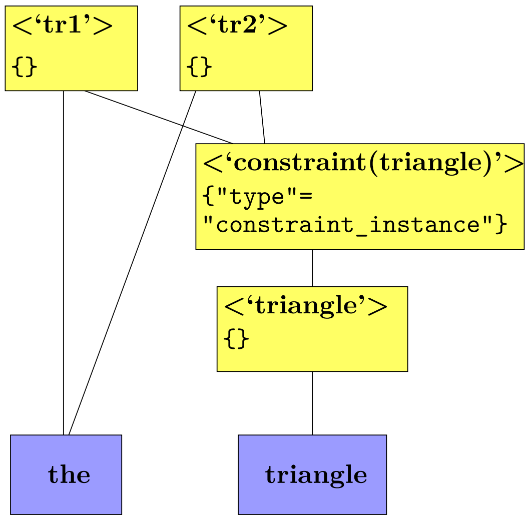
the most important one being a concept. Every node contains a concept which is the meaning of that node. An exception to this is the list of input words, which is also converted into a series of parse tree nodes to form the bottom layer of the parse tree.

Besides a concept, each node also contains a set of syntactic features represented as key-value pairs. Furthermore a node also contains information on for which input words it is a parse. Lets have a look at an example.

Figure 5.4 contains an example parse tree. I use the notation “<‘square’>” to indicate that a parse node includes a reference to the concept named “square” in the concept graph. The curly braces indicate the syntactic features carried by a parse node. The parse tree in this figure was generated by three pattern matchers, against the concept graph of Figure 5.2. The first one recognizes the word “square” and creates a parse tree node with the ‘square’ concept in it. A second recognizer recognizes parse tree nodes that contain instances of ‘kategory’ or concepts that are a subtype of such instances. If it finds one it creates a new parse tree node that contains a newly created concept, for this example I have named the new node “constraint(square)”. This new concept has in the concept graph an ‘instanceof’ relation with ‘square’, because it represents the constraint of being a square. Such a concept is a temporary one, that is created by the parser and that should be removed if it is no longer useful, i.e. after the parse is completed and the result processed. The role of this concept as a constraint is marked by the syntactic feature ‘type=constraint_instance’.[4]

[4]In a larger concept graph the status of such a concept can also be represented as e.g. a special kind of ‘istype’ concept relation to a ‘constraint’ concept. In the graph of

Figure 5.5: Parse Example (“the triangle”)



A third recognizer recognizes a definite article and a constraint.[5] It then produces parse nodes for every (non-temporary) concept in the concept graph that matches this constraint. If there is more than one match, the parse tree has multiple interpretations, as shown in Figure 5.5.[6] How to disambiguate them will be discussed in the following section.

It may seem unnecessarily complex to use both relations in the graph and syntactic features to model information. This is needed because a single concept can be referred to in different ways, and such different expressions can have different syntactic features, such as gender. For instance, the Dutch phrases “de driehoek” (“the triangle”) and “het ding” (“the thing”) may both refer to the same entity, but “driehoek” in dutch has male gender while “ding” has neutral gender. In the parse tree as described here both phrases would be interpreted as the same concept, so features such as gender can not be properties of a concept.

Figure 5.2 this is not implemented, but it is in the implementation described in chapter 6.

[5]In the implementation program described in chapter 6, which works with Dutch input, the equivalent pattern matcher also has to ensure the gender of the article and constraint node match. To allow this the parse nodes carry an extra syntactic feature for “gender”.

[6]So, the parse ‘tree’ is not in fact strictly a tree, but a directed acyclic graph (DAG). A DAG such as this one can be seen as a set of trees, where the trees may share nodes. I will continue referring to the result of the parser as a tree since in the end we are only interested in one tree contained in the DAG.

5.3 Disambiguation

The parsing algorithm as described above does not disambiguate ambiguous phrases such as Figure 5.5. The way I will do this is by attaching an extra piece of information to each node in the parse tree, a probability value. This is a real number between 0 and 1, which indicates how likely it is that the node it is part of is the best parse for a phrase. The absolute value for a node probability is not important, only its value relative to other parse nodes that cover the same set of input words. In other words, the probability is only used to pick the best node in case of ambiguity.

To calculate such probability values, we need two pieces of information. The first a prior probability of a certain concept being used, and the second one is how the words that are used in an input phrase interact with this prior probability.

The second one, the information on how input words interact with the prior probability, only plays a role during the parsing of a single sentence, phrase, or other entity that can be parsed as a whole. The prior probability, on the other hand, can depend on a lot of factors. It is in this prior probability that most contextual information comes in.

5.3.1 Activation

The prior probability can be broken down into at least two factors. The first one is the probability in isolation that a certain concept will be used in a parse tree. This value can thus be modeled as a single number for each concept. Such values can be said to correspond roughly to the salience of objects in a certain situation. As such, these values differ from one situation to the next. I will call this value the activation value of a concept.

Both the concept graph and the corresponding activation values are stipulated to be implemented in the brain. Since the probability of certain concepts being used in conversation depends heavily on the situation a person and thus his brain finds himself in, the activation values should change depending on the context. For example, if there is a conversation going on about a certain topic, then concepts that correspond to things that are related to this topic would have a high probability of being used in a parse tree. If the conversation shifts to a different topic, different concepts would become more likely. Similar changes should happen depending on the physical environment someone finds himself in, and probably a number of other contextual factors can be important as well. I will come back to this in later sections.

5.3.2 Applicability Factors

The second factor is how well certain concepts can be combined in a relation. Some relations can be applied to certain types of objects, for example the relation of being a color in the graph from Figure 5.2 can apply to instances of 'object', but not to e.g. points in time 't1' and 't2'. In that case this factor is binary: some concepts can have a color relation and some can not, but in general this should be modeled as a continuous value. I will call this value an 'applicability factor'.

If we look back at the "cut the grass" example from subsection 2.3.2, we can model the situation as there being two kinds of cutting operations, both represented as concepts, one for cutting as it is done with grass sods, and one for mowing grass. We can model 'grass' and 'grass sod' as two different concepts, both subtypes of 'object'. Then, the likelihood of the 'mow' operation being applied to instances of 'grass' is much higher than the likelihood of it being applied to instances of 'grass sod', so the applicability factor of 'mow' applied to 'grass' is larger than that of 'mow' applied to 'grass sod'. Similarly the 'grass sod cutting' operation applied to instances of 'grass sod' has a higher applicability factor than it applied to instances of 'grass'

Such applicability factors can exist for classes of things in general, such as 'grass' and 'grass sod' as in the previous paragraph, but it can also apply to specific objects. For instance, if you know that your car has a large tendency to stall, the probability of the concept that represents stalling being applied to the concept that represents your car is higher than that concept of stalling being applied to other instances of the 'car' concept.

I should also note that while in the "cutting the grass" example only the direct object of the cutting verb is discussed, in general applicability factors can also be about the subject or combinations of subject, operation, and object.

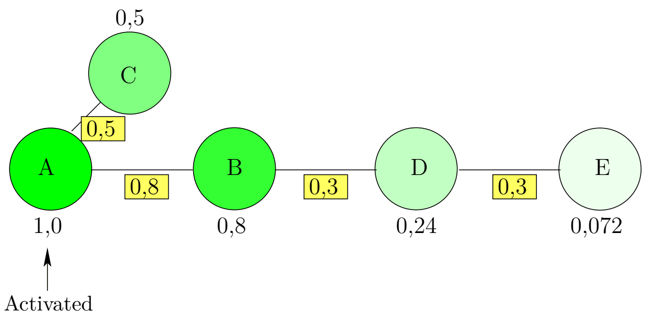
5.3.3 Relatedness Weights

In subsection 5.3.1 I have discussed what activation is supposed to represent, and noted that it should change depending on the situation. I will now have a look at some ways in which this activation can change.

The simplest way is that if a certain concept is used in a conversation, it is likely to be used again. So, after a concept is used as part of the interpretation that is selected as the best parse of an input phrase, its activation should be boosted. But the same holds true for objects that are in our surroundings, so if a particular object is seen or sensed in some other way the concept that represents it should receive some extra activation.

A concept is also more likely to be used in a conversation if it is closely related to a concept that has just been used. So, concepts should receive

Figure 5.6: Spreading of Activation Example



extra activation if a concept they are closely related to gets a higher activation. For example, if we talk about boats, the chance of sails coming up in the conversation is much larger than if we are talking about mountain climbing.

This can be modeled by having weights associated with pairs of concepts. Pairs of concepts that are closely related have high such weights, and concepts that are not have low weights. These weights can also be represented as real numbers between 0 and 1. I will name these weights ‘relatedness weights’. If one concept is activated, other concepts receive an amount of activation relative to the value of the relatedness weight they have with the activated concept. These concepts that receive such secondary activation then again cause extra activation in concepts that are related to them, and so on. So the activation spreads through the concept nodes. Since the weights are numbers between 0 and 1 and the activation a node receives is proportional to the weight, the activation a related concept receives is always smaller than that what the first concept received. As such, the amount of activation that is added dies out the further it spreads through indirectly related concepts.[7]

Figure 5.6 displays a simple example of how a simple case of activation spreading works. The nodes in this example are nodes in the concept graph, the edges that are displayed are not concept relations, but indicate that there is a relatedness weight between the two concepts. Concept A is activated to a value of 1.0, this activation is spread through the network multiplied by the weights that are displayed next to the edges. As the nodes are further away from the originally activated node, their activation dies out quickly.[8]

[7]Supporting evidence that the brain does something like this can be found in priming effects, for instance when a test subject is exposed to the word “table” he or she will be able to identify the word “chair” more quickly. See also [Reisberg, 2007, p. 255, 517].

[8]This example does not deal with the case where a circular path exists in the graph. In such a case one node can receive activation from several nodes with which it has a relatedness weight higher than 0. There are different ways in which details such as this could be handled, the best choice would need to be a matter of additional research. In

Modeled in this way, relatedness of concepts for the purpose of activation spreading can be modeled independently of the concept relations these concepts have in the concept graph. Most of the time pairs of concepts that have a relatedness weight of more than 0 will also have some kind of relation in the concept graph, but these two things should be distinguished conceptually and they do not necessarily need to coincide.

If a concept is not used again in a conversation for some time, or if an object that a concept represents is no longer present in ones surrounding, the activation of these concepts should be lowered again. This can be implemented by having the activation of concepts fall gradually over time. If the concepts often receive new activation by being used or because they have high relation weights to concepts that are used, they will on average maintain a higher activation value than concepts that are not used often.

5.3.4 Contrast Sets

One final factor that can influences the salience of specifically relations is the contrast set of things that are currently under consideration.[9] If people are presented with a group of items, they will focus on the things in which they differ and tend to ignore ways in which they are equal. Such behavior can be implemented in the current model by boosting the activation of relation concepts if a number of objects with high activations all have this type of relation with another concept, and if the targets of these relations tend to differ.

If we go back to the graph in Figure 5.2, if there would be more instances of squares and triangles, and if many of them had a high activation, and if those with high activation had many different colors, the ‘iscolor’ concept would be activated more. If, on the other hand, most such objects with a high activation had the same color, the ‘iscolor’ concept would not receive extra activation.

5.3.5 Other Influences on Activation

It could be that there are other factors important in how the activation changes that can not be captured by the mechanisms described above. If so, that may be topics for future research. The mechanisms described here seem to be sufficient for many often used examples.

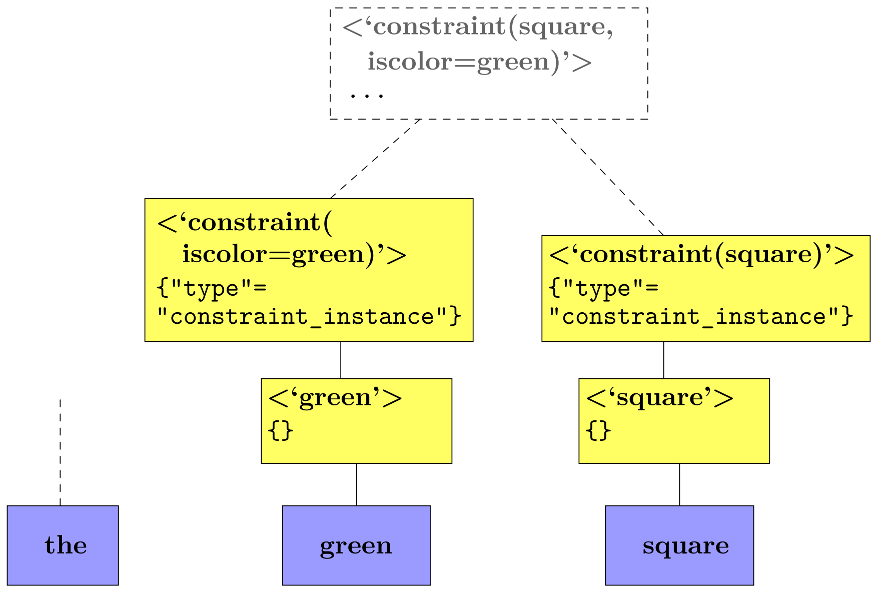
5.3.6 Usage in the Parser

Now that I have given a description of what mechanisms should determine the brain’s model of the prior probability of a given concept being used in

section 6.4, where I describe the implemented program, I give a more detailed algorithm for one way to handle this.

[9] [Tversky, 1997]

Figure 5.7: A parse tree being built



<'constraint(square, iscolor=green)'>
...
<'constraint(iscolor=green)'>
{"type"="constraint_instance"}
<'constraint(square)'>
{"type"="constraint_instance"}
<'green'>
{}
<'square'>
{}
the
green
square

language, we will move to how this information can be used to disambiguate multiple possible parses.

As stated before, each parse node in the parse tree (or DAG) contains among other things a probability value between 0 and 1. This probability value is calculated based on the activation values and other weights discussed above. The exact way in which such variables influence the probability value of a parse node depends on the pattern matcher that generates the node. In general, the probability value of a parse node corresponds to the activation value of the concept in the node. If a parse node has two parents that both contain concepts, the activation value of the node will also take the probabilities of the parent nodes into account. For pattern matchers that for example construct constraint concepts in which a certain property is applied to an instance of a certain object type, the probability depends on the previous factors and also on the applicability factor of the property.

I will illustrate that last case with an example. In Figure 5.7 we see a parse tree being built for the phrase "the green square". Nodes have been constructed that represent the constraint of having the 'iscolor' relation to the 'green' concept, and the constraint of being an instance of 'square'. The probability values of these two nodes depend on the activations of the 'iscolor' and 'green' concepts and 'instanceof' and 'square' concepts respectively. The next thing that will be done is constructing a parse node that contains a combined constraint of being a green square. The probability of this node will be calculated based on the probabilities of the other two constraint nodes, which become its children, and the applicability factor of having an ‘iscolor’ relation between instances of ‘square’ and instances of ‘color’. (Unless there is a more specific applicability factor for having the ‘iscolor’ relation between the concepts of ‘green’ and instances of ‘square’, but we will suppose there are not.) While this example shows the combination of an adjective and a noun, the same thing happens if a verb is applied to a (phrase denoting an) object.

5.3.7 Calculation of Weights

In the above description I have not specified any concrete formulas for how to combine different values or weights. The simplest way to do that is to just multiply the values together. Since all values are real numbers between 0 and 1, the result will also be a number between 0 and 1. This may seem to have the disadvantage that the resulting probability values can get very small if they are the result of a large number of multiplications, but since the value of these probabilities only matters in relation to the probabilities of other parse nodes this should not matter because the other values should be the result of a similar number of multiplications.

The choice for doing simple multiplication is a bit tentative. Given a lack of specific evidence for a specific kind of combination function multiplication wins on grounds of simplicity, but if evidence for a specific kind of combination function became available this model should probably be updated accordingly.

5.4 An Example

I will now consider a more complex example in more detail, the example of “cut the grass” that has been discussed before in subsection 2.3.2 as part of the context shifting argument.

Any kind of relevant context will need to be represented in the concept graph. We will suppose a concept graph that contains at least concepts for ‘grass field’, ‘grass sod’, ‘mow’, ‘grass-sod-cut’, and instances of ‘grass field’ and ‘grass sod’. I will name the instances of ‘grass field’ and ‘grass sod’ as ‘f1’, ‘f2’, etc. for the fields and ‘s1’, ‘s2’, etc. for the sods. The graph additionally contains concepts of ‘imperative-action’ and two relations ‘action-type’ and ‘action-target’. The concept graph is stipulated to represent everything that someone knows (at least explicitly), therefore everything that someone knows about grass fields and grass sod should also be encoded in the concept graph. Real world things like grass are very complicated to describe completely in a mathematical sense, so it is not possible to specify a full conceptual graph of everything that is related to ‘grass field’ or ‘grass sod’ in some way in a paper like this. So for this example we will focus on only a small subset of this hypothetical graph, which are the concepts I just mentioned.

We also need a few pattern matchers to form a parser for this example we will use:

- A pattern matcher that matches the verb “cut”. It produces two parse nodes (as far as we care about in this example), one containing the ‘mow’ concept and another containing the ‘grass-sod-cut’ concept. Both nodes contain syntactic features to indicate that they are imperatives. The probability of both parse nodes is equal to the activation of the contained concepts.

- A pattern matcher that matches the word “grass”. Like the one for “cut”, this matcher produces two nodes we care about. The concepts in both nodes are new temporary concepts that encode the constraints of being an instance of ‘grass field’ and ‘grass sod’ respectively. Both nodes have a syntactic feature to mark them as constraints. The probability of the nodes equals the activation of the ‘grass field’ and ‘grass sod’ concepts.[10]

- A pattern matcher that matches the word “the”, and a constraint concept. The results produced by this pattern matcher are nodes for all concepts in the concept graph that match the constraint. The probability of the node is the activation of the contained concept multiplied by the probability of the matched constraint node.

- A pattern matcher that matches first a node that contain an action concept and that has the syntactic feature indicating it is an imperative, and second a node that contains a kind of concept to which the action in its first match can be applied. Whether an action is applicable is determined from the set of applicability factors. The applicability factor of the action to the concept must be larger than 0, or the applicability factor of the action to something that is of the type that the concept has (as indicated by its ‘instanceof’ relation) must be larger than 0.

  The output node contains a newly constructed concept that is an instance of ‘imperative-action’. It has relation ‘action-type’ with the concept in the matcher’s left match, and ‘action-target’ with the concept in the matcher’s right match. The probability is the product of the applicability factor and the probabilities of both child parse nodes.

[10]Compared to the parse in Figure 5.4 and Figure 5.5 on page 37, I have contracted the matchers that first recognize an instance of ‘kategory’ and then produce a constraint concept from it into a single matcher, to be a bit simpler. What choice is made depends on the exact definition of the grammar that should be recognized.

> If we also wanted to model something about who an imperative sentence is spoken to, we would need to extend the concepts we are interested in to include concepts that represent persons, and a 'action-subject' relation. This pattern matcher should then also create an 'action-subject' relation for the newly created 'imperative-action' instance, that has the person spoken to as target. In the interest of simplicity we will skip over this part and only discuss the grass and the cutting.

To access any relations within the conceptual graph, it needs to be known which instance of 'world' needs to be searched. By default, this will be the present actual world. Since we are always in the actual world we can assume that the actual world usually has the highest activation of all worlds. Exceptions can occur if for example we are specifically talking about the past or about a fictional world. In the "cut the grass" example, there is no such special context, so all analysis can be done on the subgraph that encodes the relations for the actual world.

Lastly, we will also need to assume some values for applicability factors. We will assume that the applicability of 'mow' to instances of 'grass field' is high, and so is that of 'grass-sod-cut' to instances of 'grass sod', let us value them at 0.9. The actions can not easily be applied the other way around, but we will assume that it is possible to use a lawn mower to mow pieces of grass sod, so we will give the applicability of 'mow' to instances of 'grass sod' a value 0.1. Cutting a grass field into slices seems much more impossible so we will set the applicability factor of 'grass-sod-cut' to instances of 'grass field' to 0.

In the same way, we can suppose that the relatedness weights between the 'mow' concept and the 'grass field' concept is relatively high, just as the one between 'grass-sod-cut' and 'grass sod', while those between 'mow' and 'grass sod' and 'grass-sod-cut' and 'grass field' are zero or very low.

5.4.1 Mowing the Lawn

We will now look at two contexts separately. The difference between both contexts, as far as this model is concerned, is in the activation of concepts. In the first case we will assume that the sentence is spoken as part of a conversation in which the garden has been featured. Because of this the concept in the listeners mind that represents the garden at hand has a high activation. There is a lawn in this garden, that is also represented in the listeners mind as the concept 'f1' and that is an instance of the 'grass field' concept. Since this lawn is part of the garden, we can assume that the relatedness weight between the two corresponding concepts also has a large value. Because of this the concept that represents the lawn and is an instance of 'grass field' also receives some activation. This in turn activates the concept of ‘grass field’ a bit, and this the concept of ‘mow’, although it is an even smaller amount of activation. However, it is still more than the ‘grass-sod-cut’ concept received.[11].

In the situation as described here, there is no grass sod as such present. We can still assume that the listener knows about a piece of grass sod somewhere not directly related to the current conversation, so there can still be an ‘s1’ concept in the listeners mind, but this concept would have a negligible activation value.

So, let us assume the following activation values in this context:

‘grass field’:	0.4
‘grass sod’:	0.2
‘mow’:	0.2
‘grass-sod-cut’:	0.05
‘f1’:	0.5
‘s1’:	0.001

Figure 5.8 shows the parse tree that results. The nodes in the tree also contain information on which words are covered by them, so only the nodes that cover the whole input sentence are considered as a final interpretation. Since the applicability factor of ‘grass-sod-cut’ to ‘f1’ is zero, no corresponding action concept and parse node is generated for that combination. The interpretation of mowing the lawn has a final probability value of 0,036, which is 2000 times larger than that of the next most probable interpretation of cutting the grass sod.

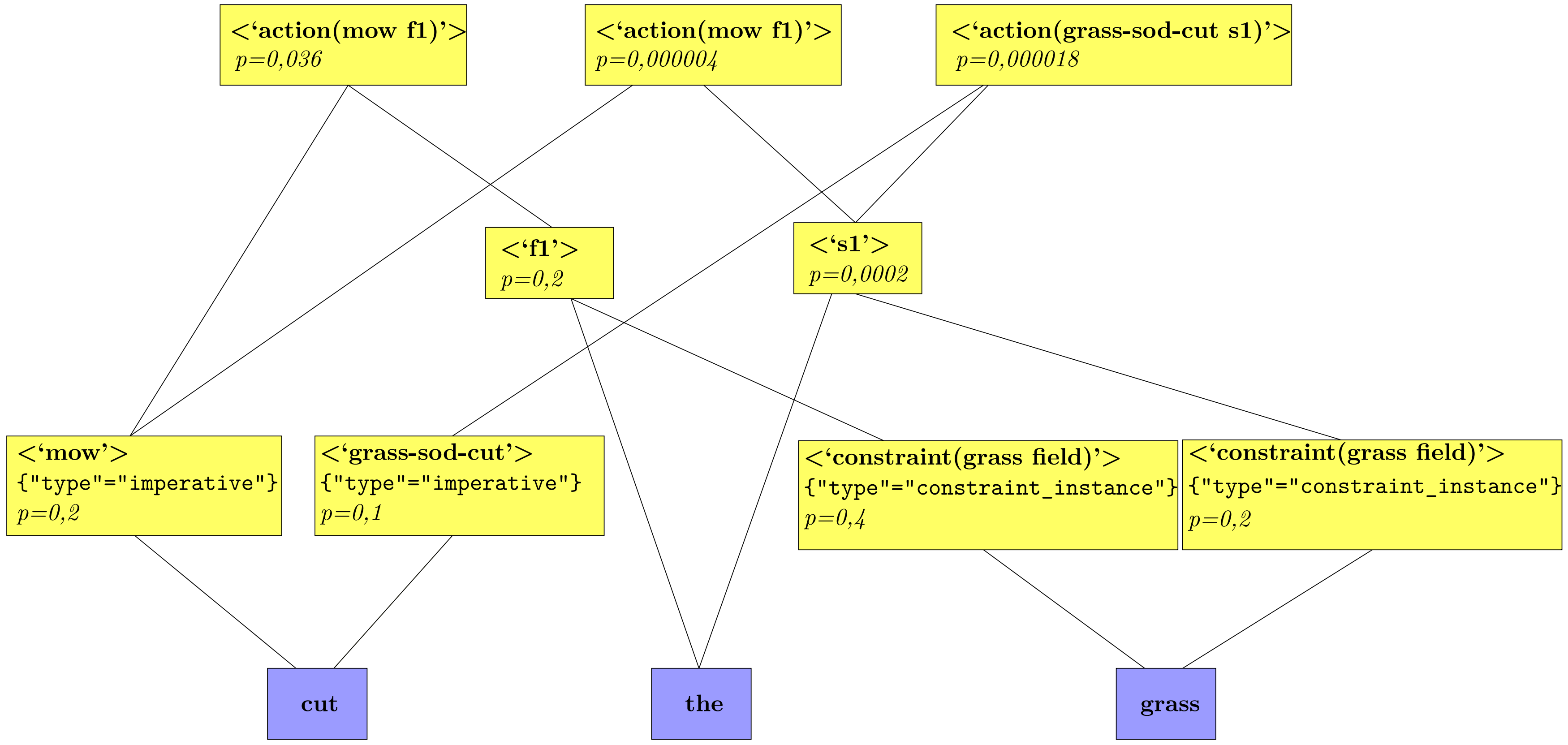
5.4.2 Cutting Grass Sod

Now let us look at the other situation, the one in the grass sod farm. We will suppose the listener is an employee who regularly cuts grass sod for customers. In this case, we can assume that the listener knows the piece of grass sod that the farm uses to takes slices of for customers. The grass sod farm itself would also be represented in the listeners mind with a relatively high activation because the listener is in it, and the relatedness of this farm and the piece of grass sod ‘s1’ would also be relatively high. Similarly, any lawns that the listener knows about would not have a very high activation value.

Let us assume the following activation values:

[11]The activation will also spread in other ways, for example the concept ‘garden’ (the concept itself, not the instance of ‘garden’ that represents the garden at hand) will also have a relatively large relatedness to the concept of ‘grass field’. For now we will not go into this in detail.

Figure 5.8: Lawn Mower Parse



‘grass field’:	0.2
‘grass sod’:	0.4
‘mow’:	0.1
‘grass-sod-cut’:	0.4
‘f1’:	0.01
‘s1’:	0.4

The resulting parse tree with probability values is shown in Figure 5.9. As can be seen, the most likely interpretation in this case is the one of the cutting-in-the-grass-sod-sense of the piece of grass sod ‘s1’.

5.4.3 Extending the example

The example as shown here works, but it could be extended to make the resulting interpretations more robust.

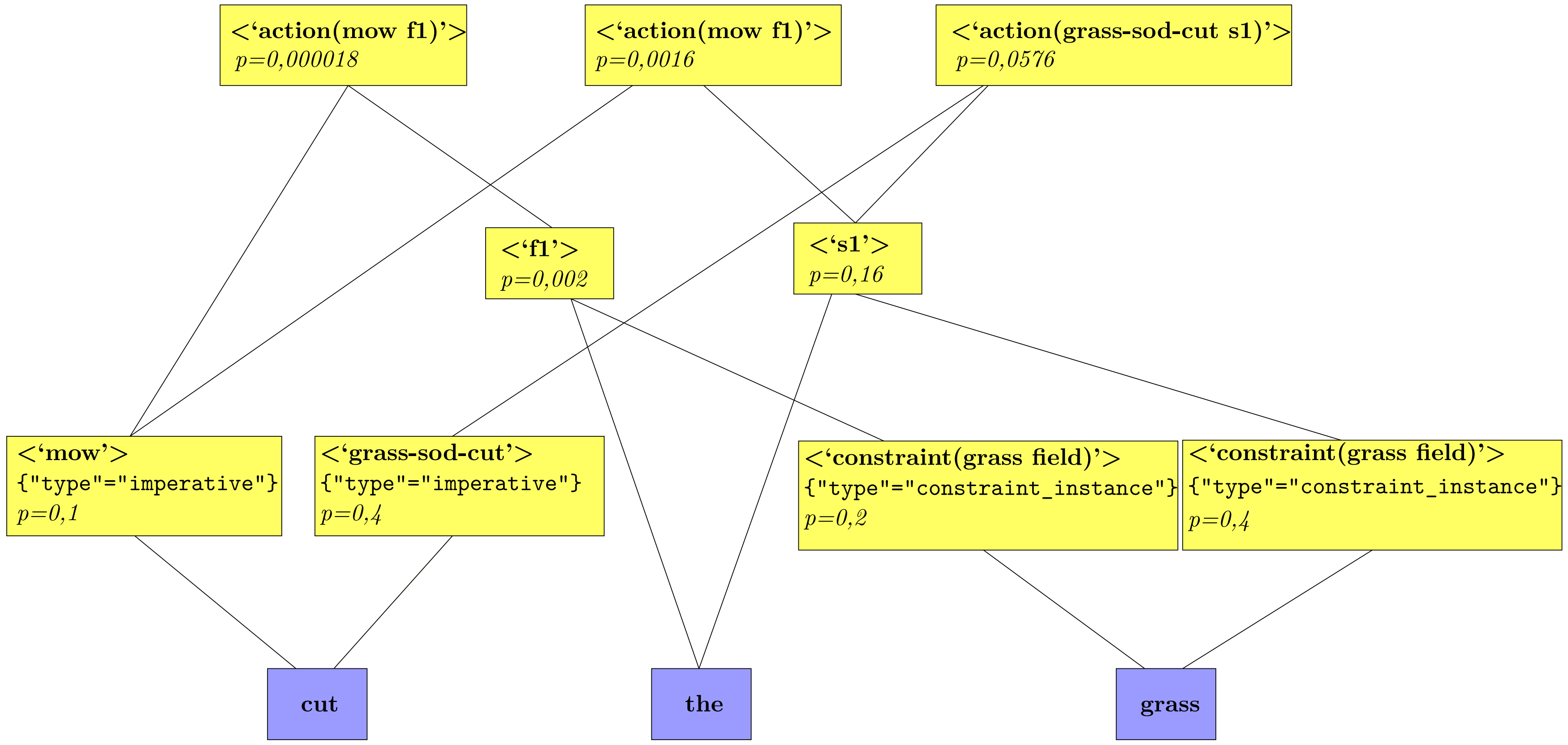
For one thing, if the listener in the second situation is not an employee, but e.g. the customer, he would not know about the piece of grass sod the speaker is referring to. In other words, he would not have concept ‘s1’ in his mind. In such a case we would need to extend the third pattern matcher so that if no suitable object that fits the matched constraint is found, where “suitable” means that it has an activation value higher than a certain minimum, the pattern matcher will create such an instance that matches the constraints itself. This instance would need to carry additional information either as syntactic features or as conceptual relations to indicate that it is an “indefinite” concept, that could still be unified with a different concept if the listener learns which piece of grass sod was being talked about. So at a higher level, what could happen in such a case is that the listener assumes that a piece of grass sod exists and that the sentence in question was referring to it.

Another way to make the example more robust is to include goals of different people in the concept graph. We could extend the concept graph with a ‘goal’ concept and a ‘hasgoal’ relation. Then, in the first situation, if the listener has inferred that the speaker has the intention of having his lawn mowed, this would be represented as a concept that is an instance of ‘goal’, and that has suitable relations with the concept representing the speaker and the concept representing the lawn[12]. Since theory of mind is an important part of human interaction[13], we can presume that instances of ‘goal’ have strong relatedness weights with the concepts they have relations with. Therefore, since the concept representing the speaker has a high activation, so will his goals and thus the concept representing the lawn. In

[12]A “suitable relation” in this case would probably not be just a single relation in the conceptual graph. This would require some representation of the desired state of the lawn, and a concept relation from the goal instance concept to this representation.

[13]Background on “theory of mind”: [Meltzoff, 2002], [Samsom, 2013]

Figure 5.9: Grass Sod Farm Parse



<‘action(mow f1)’> p=0,000018
<‘action(mow f1)’> p=0,0016
<‘action(grass-sod-cut s1)’> p=0,0576
<‘f1’> p=0,002
<‘s1’> p=0,16
<‘mow’> {"type"="imperative"} p=0,1
<‘grass-sod-cut’> {"type"="imperative"} p=0,4
<‘constraint(grass field)’> {"type"="constraint_instance"} p=0,2
<‘constraint(grass field)’> {"type"="constraint_instance"} p=0,4
cut
the
grass

the case where the sentence is used within the sod farm, the listener would know that the speaker has a goal of giving the customer a piece of grass sod. Therefore, like in the mowing the lawn case, the concept representing the speaker in the listener's mind will have a high activation, and so will the concept representing this goal, and thus the concept of 'grass sod'.

Chapter 6

Implementation

In order to confirm that the DPS model as described previously can actually work I have written a program that implements the DPS model. The program also serves to explore possible ways in which aspects of the DPS model that were not defined in complete detail above could be implemented. The program is a proof of concept. It is named (rather uncreatively) 'DPSP', for 'Direct Probabilistic Semantic Parser'.

The program does not implement the full DPS model due to time limitations. For the conceptual graph, the program does not implement world-relative relations. As such, the program can only represent a single state of the world at a time.

The concept graph that the program contains has concepts that allow it to describe a small 2-dimensional scene with geometric objects in it. The program displays the state of this scene on the computer screen as it is described by its concept graph. It is also possible to input sentences, and the program will parse them into concepts of its concept graph. If the sentences are of certain recognized imperative forms, the program will respond to them by changing the conceptual graph in the way that is asked. However, since the concept graph of the program only represents one state of the world, it does not remember any such changes that were made. It can therefore not interpret sentences that are about any past or any actions it has done. Because of this, we can say that the program only understands the present of its virtual world.

While the program is written in English, the input sentences it understands are in Dutch. I will provide English translations of any example inputs I use in this document.

The program also does not implement all disambiguation mechanisms that are described in the previous chapter. It implements activation of concepts and relatedness weights, but not applicability factors or contrast sets. Since the program has no senses, or any notion of an "outside world", the only way in which concept activation is increased is if a concept is used

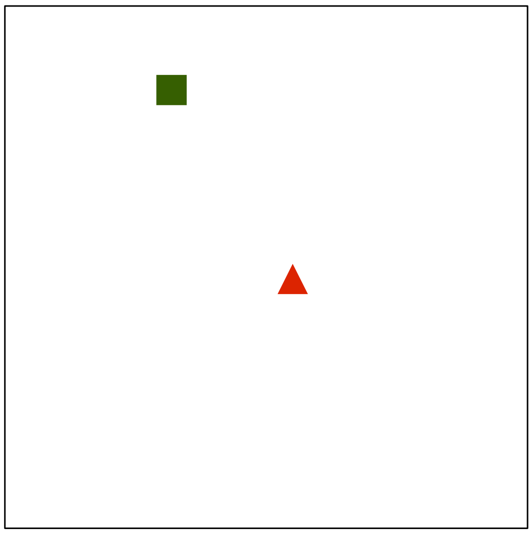


Figure 6.1: The scene when the program starts up, containing only two shapes

in the interpretation of an input sentence.

This chapter only describes how the DPS model is implemented in the program. For more details on the kinds of commands the program can interpret and other capabilities see Appendix A.

6.1 Concept Graph

The DPSP program contains a graph of concepts. The concepts themselves are the nodes in the graph. They are taken to be atoms that do not carry any information other than their identity and the relations they have with other concepts. The relations are encoded in a set of 3-tuples, which is in contrast to the graph in the DPS model above that uses 4-tuples. The tuples are $\langle type, subject, target \rangle$, where each of *type*, *subject*, and *target* are concepts. Thus, the concept graph only represents a single state of the world. The concept graph can be defined as

$$\langle C, R \rangle$$

where

$$R \subseteq \{\langle t, s, g \rangle \mid t, s, g \in C\}$$

Similar to the description of the DPS model, each concept in the program can carry a text name. This name is not used in any algorithm but only serves to make the program easier to understand and debug.

Since the program does not have any sensors to observe the world around it, I have implemented an internal virtual world that consists of a simple two dimensional scene on which a few simple geometric objects are placed. We could say metaphorically that the scene that the program displays shows a world that the program is thinking about, an imaginary world that exists in the programs imagination. The default scene the program starts with is shown in figure 6.1.

Some concepts that are currently created at program startup include ones with the following names:

object	red	property
kategory	green	iscolor
square	yellow	isat
triangle	blue	vk1
location	subtypeof	dr1
color	instanceof	

The following tuples are part of the concept graph:

$\langle \text{subtypeof}, \text{square}, \text{object} \rangle$
$\langle \text{subtypeof}, \text{triangle}, \text{object} \rangle$

$\langle \text{instanceof}, \text{object}, \text{kategory} \rangle$
$\langle \text{instanceof}, \text{square}, \text{kategory} \rangle$
$\langle \text{instanceof}, \text{triangle}, \text{kategory} \rangle$
$\langle \text{instanceof}, \text{property}, \text{kategory} \rangle$
$\langle \text{instanceof}, \text{kategory}, \text{kategory} \rangle$
$\langle \text{instanceof}, \text{color}, \text{kategory} \rangle$
$\langle \text{instanceof}, \text{location}, \text{kategory} \rangle$

$\langle \text{instanceof}, \text{vk1}, \text{square} \rangle$
$\langle \text{instanceof}, \text{dr1}, \text{triangle} \rangle$

$\langle \text{instanceof}, \text{red}, \text{color} \rangle$
$\langle \text{instanceof}, \text{green}, \text{color} \rangle$
$\langle \text{instanceof}, \text{yellow}, \text{color} \rangle$
$\langle \text{instanceof}, \text{blue}, \text{color} \rangle$

$\langle \text{instanceof}, \text{iscolor}, \text{property} \rangle$
$\langle \text{instanceof}, \text{isat}, \text{property} \rangle$
$\langle \text{instanceof}, \text{subtypeof}, \text{property} \rangle$
$\langle \text{instanceof}, \text{instanceof}, \text{property} \rangle$

$\langle \text{iscolor}, \text{vk1}, \text{green} \rangle$
$\langle \text{iscolor}, \text{dr1}, \text{red} \rangle$

The set of tuples encodes multiple relations, indicated by the first element of the tuple. The above list contains the 'subtypeof' relation, the 'instanceof' relation and the 'iscolor' relation. The relations themselves are thus identified with concepts. In the rest of this chapter I will speak of the set of 3-tuples R as containing the relations of the concept graph, and I will speak of concepts that appear as the first element in such a tuple as being a relation, in the same way as I described the DPS graph model.

Every concept that is also a relation is an instance of 'property' or a subtype of 'property',[1] as indicated by the 'instanceof' and 'subtypeof' relations, and every concept that is an instance of 'property' or one of its

[1]This is a slight change in terminology from the DPS model, where I used the name

subtypes is a relation. The above list therefore contains a fourth relation, ‘isat‘, that is empty according to this list.

In the above list it can be seen that ‘vk1’ is in the ‘instanceof’ relation with the ‘square’ concept. I will refer to this as ‘vk1’ being a ‘square’ or as ‘vk1’ being an instance of ‘square’.

The above lists of concepts and relations is not the full concept graph as currently implemented. Among other things, the program also contains a number of instances of ‘location’, each of which represents a point in a 2d space. Both ‘vk1’ and ‘dr1’ have an ‘isat’ relation with one of these instances of ‘location’.

The program also contains recursive versions of the ‘subtypeof’ and ‘instanceof’ relations, so that for example ‘vk1’ and ‘dr1’ are in this recursive instanceof relation with ‘object’.

The concept graph is implemented in the file `dpsp.py`.

6.1.1 What the graph represents

The concept graph in the DPS model as it is postulated to exist in the brain represents the model the brain has of its surrounding environment. As such a concept graph should be able to represent positive facts, but also negative ones or facts where a certain variable is known to have one of a set of values, but not others. Ideally the graph should also encode things like how certain or reliable a piece of information is.

The concept graph as implemented in this program does not do all of this. It can only encode positive relations, but no negations or other kinds of facts. In the current implementation this is not a problem, the program simply does not understand any kind of negated language. Knowledge of which relations can only be instantiated once per object is built in, so for example changing the color of an object simply replaces the relation tuple which encodes the color for that object with a different one. (The current program only allows an object to have one color.)

6.2 Parser Algorithm

The parser is based on a standard bottom up chart parser. The input to the parser is a list of words. The parser creates a table of size $n * n$, where n is the length of the input list. Each cell in the table at height h will contain the parse tree nodes that represent a possible parse of the next h words.

Like in the DPS model, the parse tree contains references to concepts. Each node of a parse tree has as data members a concept and a number of syntactical features stored as key-value pairs.

“relation” for this concept.

Table 6.1 shows an example of a simple parse of the phrase “het vierkant” (“the square”).

2	<“vk1”> {`gender=n`}	

1		<“newinstance”> {`gender=n`, `constraint_instance=True`}
		<“square”> {}

	het	vierkant

Table 6.1: Simple parse example

The height of each cell indicates for how many consecutive words it contains possible interpretations. So the cells on the bottom row contain interpretations for just the single word over which they are placed, and the cell with height two over “het” contains interpretations for the word over which it is positioned and the next word. (So in this example that is the entire phrase). The lower right cell, which contains interpretations for the word “vierkant”, has two concepts in it so there are two possible interpretations. The notation <“square”> indicates the concept named “square”. This is the concept ‘square’ from the conceptual graph described above, and represents an interpretation where the word “vierkant” is interpreted to refer to the shape ‘square’, i.e. not to a specific square object.

The second concept is named “newinstance”. It is not listed in the conceptual graph above, as it is created by the parsing algorithm and added to the concept graph as a temporary concept. Although I have not displayed it in the figure, this concept has an ‘instanceof’ relation to ‘square’ that was also created by the parsing algorithm. This concept represents an as of yet unknown reference to something that as a constraint must have an ‘instanceof’ relation to ‘square’. The parse node consists of this new concept and a set of syntactic features that indicate the gender that this concept has here (neutral), and that it is an as of yet undefined reference with constraints.

In the upper left cell we find the concept ‘vk1’, which is again a concept from the conceptual graph above. As this cell is at height two, it contains interpretations for the whole two-word phrase. As there is only one square (i.e. instance of the ‘square’ concept) in the conceptual graph there is only one possible interpretation for this phrase.

6.3 Language Rules

The language is described by a set of rules similar to production rules in a BNF grammar. Each rule takes one or more elements as input and produces one or more outputs. The inputs are either parse tree nodes that contain a concept and syntax features, or words from the input (possibly enriched with syntax features). The outputs are parse nodes consisting of a concept and optionally syntactic features. Each parser rule specifies a set of features that its inputs are required to have. These features can be either syntactic features or conceptual relations. The current implementation uses program code to calculate the output(s) based on the input and the concept graph. The program code is not very difficult so I have just printed it here rather than using a formalism such as lambda calculus.

The parser rules that are involved in the example from Table 6.1 are shown in listing 6.1 on this page.

Listing 6.1: Language rules for the example in Table 6.1.

```
@parser({'word': 'vierkant'})
def SquareRec(pcontext):
  yield C.square, {'gender':'o'}

@parser({instanceof:kategory})
def KategoryConstraintRec(pcontext):
  kat = pcontext.match[0].concept
  yield pcontext.transientconcept(
            "constraint({})".format(kat),
            {instanceof:kat, istype:indeterminate}),
        dict(pcontext.match[0].features,
             constraint_instance=True)

@parser({'determiner':True, 'definite':True, 'indexical':None},
        {'constraint_instance':True})
def InstanceRec(pcontext):
  g1, g2 = (m.features['gender'] for m in pcontext.match)
  if g1 != g2 and g1 != None:
    return
  for c in
      pcontext.match[1].concept.instanceof.fullextension():
    if not c.testrel(istype, indeterminate)
       and constraintmatch(c,
             constraint=pcontext.match[1].concept):
      yield c, {}, activation[c]
```

Each rule consists of a specification of the input that matches it and a function that produces the output. The input specification is described in the decorator `@parser(...)`. Each dictionary denoted by `{...}` specifies conditions for one input. String keys (surrounded by quote marks) are syntactical features, keys that are not strings describe concept relations. So in listing 6.1 we see that `SquareRec` requires one input with a syntactic feature 'word' to have the value 'vierkant'. The syntactic feature 'word' denotes a word from the input list of words. `KategoryConstraintRec` takes one input that has to be a concept that is an instance of 'kategory'. The `InstanceRec` rule takes two inputs, the first having a number of syntactic features such as being a definite determiner but not a demonstrative, and the second one as being a constraint representation.

The output functions (signified by the `def` keyword) in each of the rules are quite simple too. Each function has a `pcontext` argument, which is an object that contains all required information about the current parse. `pcontext.match` contains a list with the matched input parse nodes.

The first parse rule unconditionally returns the 'square' concept with the syntactic feature of having neutral gender. The `KategoryConstraintRec` creates a temporary concept that represents a constraint. The expression `"constraint({})".format(kat)` on line 9 creates the name for this new concept and is not of algorithmic importance. The new concept gets the relations of being an instance of the input concept, and gets an 'istype' relation 'indeterminate'. 'istype' and 'indeterminate' are concepts that were not listed previously, they are used to describe special concepts such as constraints which do not in themselves represent existing things. The output also has the syntactic feature `"constraint_instance"` set to true.

The `InstanceRec` rule first checks that the genders of its two inputs match and then outputs all existing concepts that match the constraint in its second input. The `activation` return value will be discussed later.

These parser elements together form a hierarchical parser. In the example, the word "vierkant" is parsed into the concept 'square' by `SquareRec`. This concept is recognized by `KategoryConstraintRec` which produces a constraint concept. As both of these concepts represent an interpretation for only the single word "vierkant" they both end up in the same cell of the table. Finally the `InstanceRec` rule matches the determiner "het" and the constraint concept produced by `KategoryConstraintRec`, producing the final output for this example.

One loose end that this description does not cover is how input words are given their respective syntactic features. What happens is that as a parsing run is initialised each input word is transformed into a syntactic parse node, which is a parse node that does not contain a concept but just contains some syntactic features. This first parse node only has one feature `word` which has the input word as value. Next, the program also defines a few syntactic parsing rules that take such a syntactic node as input and produce another syntactic node that has additional syntactic features, such as `` `determiner':True ``. All these syntactic parse rules only take one input and produce one output, so they do not take part in any composition or other interesting things during the parse. The syntactic parse nodes are also stored in the table and are processed by the parsing algorithm in the same manner as the full parse nodes that contain concepts. The full table for the example in Table 6.1 therefore contains two additional nodes in the lower left cell, one containing just `` `word':`het' `` and the second containing additionally `` `determiner':True `` and some similar features, and one extra node in the lower right cell that contains only `` `word':`vierkant' ``.

As this description shows, this implementation does not make a qualitative distinction between lexicon and grammar. The lexicon is just a set of grammar rules that take a single word as their only input.

In the implementation parse nodes carry some more information that is not shown here. This information includes references to the parser rule by which they were generated and the inputs used, and information needed to manage the used computer resources.

Table 6.2 contains the parse grid for a larger example. This table also includes descriptions of the child nodes for each parse node, which were not shown in the smaller example in Table 6.1. These relations are displayed as a list of co-ordinates surrounded by square brackets. The co-ordinates are in columnno.–height format. If there are multiple parse nodes in one cell, the number of the intended parse node is appended to the co-ordinates with a colon. This number is counted starting from the bottom node as printed in the graph.

6.4 Disambiguation and Activation

The program implements some of the disambiguation methods as described in the previous chapter. The first one is concept activation. Each concept in the concept graph has an associated activation, which is a real number between 0 and 1. Every time a concept is "used", its activation is raised, and so is the activation of concepts that are closely related to it. Over time this activation decays again towards zero.

The parse nodes that the parser generates also carry a probability value, like in the DPS model. This probability represents the relative likelihood of that interpretation being the right one. The probability of each node depends on the probability of its children in the parse tree, and of the activation of the concept that it references. If there is more than one candidate for the final interpretation of a sentence the program chooses the one with the highest probability.

The decay of activation is exponential and based on wall clock time. The current implementation uses a decay factor so that an activation is down to one percent of its original value after three minutes.

A concept is considered to be "used" if it is the interpretation with the highest probability for an input sentence. This concept's activation is increased to 1.0 immediately when the parse is finished. The program then uses a spreading activation algorithm[2] to spread this activation to

[2]For references on spreading activation algorithms see e.g. [Aswath et al., 2005], [Anderson, 1983] and [Collins and Loftus, 1975]. See also http://en.wikipedia.org/wiki/Spreading_activation (accessed july 2013). Other choices for the details of such an algorithm could be made, but the one I use here seems to be sufficient. For other choices I

Table 6.2: Larger parsing example

5	<“changecolor(vk1, yellow)”> [(1,1) (2,3) (5,1:3)] {}				
4					
3		<“vk1”> [(2,1:2) (3,2)] {}			
2			<“combined_constraint(square, color=green)”> [(3,1:2) (4,1:2)] {'constraint_instance'=True, 'gender'='o'}		
1	SyntaxNode {'word'='maak'}	SyntaxNode [(2,1:1)] {'origword'='het', 'determiner'=True, 'definite'=True, 'gender'='o'} Syntax('het') {'word'='het'}	<“constraint(color=green)”> [(3,1:1)] {'adjectival'=True} SyntaxNode {'word'='groene'}	<“constraint(square)”> [(4,1:2)] {'constraint_instance'=True, 'gender'='o'} <“square”> [(4,1:1)] {} SyntaxNode {'word'='vierkant'}	<“color(yellow)”> [(5,1:1)] {'adjectival'=True} <“yellow”> [(5,1)] {'determined'=True} SyntaxNode {'word'='geel'}
	maak	het	groene	vierkant	geel

Algorithm 1 The activation algorithm pseudocode

```
  ▷ $c$ is the concept, $a$ the additional activation, $path$ is used to prevent
  following loops in the graph and is the empty set on the first call.
procedure ACTIVATE($c, a, path$)
    if $c \in path$ or $|path| > recursionlimit$ then
        return                         ▷ check for loops and recursion depth
    end if
    $a \leftarrow a * conceptweight[c]$          ▷ apply per-concept weight
    $activation[c] \leftarrow activation[c] + a - activation[c] * a$ ▷ update activation

    for $t \in \{t \mid \langle t, c, \_\rangle \in R \lor \langle t, \_, c\rangle \in R\}$ do
                                       ▷ activate the relation concepts
        ACTIVATE($t,\ a * relationweight[t],\ path \cup \{c\}$)
    end for
    for $\langle t, g\rangle \in \{\langle t, g\rangle \mid \langle t, c, g\rangle \in R\}$ do   ▷ activate related concepts
        ACTIVATE($g,\ a * targetweight[t],\ path \cup \{c\}$)
    end for
    for $\langle t, s\rangle \in \{\langle t, s\rangle \mid \langle t, s, c\rangle \in R\}$ do ▷ activate reverse related concepts
        ACTIVATE($s,\ a * subjectweight[t],\ path \cup \{c\}$)
    end for
end procedure
```

other related concepts. New and existing activations are combined using the formula for non-mutually exclusive 'or' of two probabilities a_1 and a_2: $activation = a_1 + a_2 - a_1 * a_2$, where a_1 is the activation a concept already has, and a_2 the added activation. The activation spreading algorithm uses a set of weights per relation type, these correspond to the relatedness weights as described by the DPS model. Each weight is a value between 0 and 1. For each relation type, there is a weight for spreading from subject to target, one for spreading from target to subject, and one for activating the relation type concept itself. Additionally, there is a weight for every concept that allows for dampening the activation that is spread through that concept. This weight is 1 by default, and a lower value for some builtin concepts such as 'kategory', because as it is the top of the type hierarchy a relation through 'kategory' does not intuitively indicate relatedness. The 'location' concept also has a very low weight, mainly because spreading activation through 'location' is not currently used but creates a big computational burden as there are so many instances of 'location' in the current implementation. In a better optimized implementation such a concept with a lot of instances might not be a problem. The algorithm is described as algorithm 1 on the current page. Note that the actual implementation does not loop over the

will refer to the literature.

entire set R but uses a more efficient implementation.

As this description shows, the weights used here are not entirely the same as those used in the DPS model. Some changes were necessary in order to ensure that propagating the activation would not take too much computing time. The program does not implement relatedness weights as entirely separate from concept relations as the DPS description does. Rather weights are associated with concept relations. Therefore activation can only be spread from one concept to another if there is a concept relation between the two. In order to compensate for this decrease in flexibility the program splits the weights into two, one weight for each direction in which the activation can spread. There is also a third weight that determines how much activation is spread to the relation concept itself.

The values of the weights would be a good target to apply machine learning algorithms to, were it not that that requires an existing training set of conversations and correct interpretations, which does not exist. The current weights were chosen based on a trial and error procedure. This can be done as long as the total concept graph is small enough. For the current implementation this seems to work.

6.5 Contextual Capabilities

The current program shows a limited amount of context dependent behavior. For example, if a specific object is referred to in an input sentence, using the word "hem" ("it") will correctly refer to that same object as long as the necessary syntactic features match. Demonstrative "die" can be used to indicate a location, an object, or an object property and the right interpretation will be selected depending on what makes sense.

A bit more advanced context sensitivity can also be handled: Assume a scene that contains multiple objects of different colors and shapes, but only one red and one green triangle. Now, if the following inputs are issued in sequence: "zet de rode driehoek daar", "zet de groene daar" ("put the red triangle there", "put the green one there"), the referential expression in the last input will be correctly interpreted as the green triangle, even though there may be other green objects in the scene.

These contextual dependencies are implemented through the activation framework as described in the previous section. Using concepts in sentences activates them and the concepts they are directly or indirectly related to. This causes interpretations of later sentences that use these activated concepts to have a higher probability. In the example above, the phrase "zet de rode driehoek daar" would activate among other things the 'triangle' concept, and through it all other instances of 'triangle'. This causes the green triangle interpretation to have the highest probability in the phrase that follows. The concepts that identify other objects would be activated as well, but they would be activated to a lesser extent since the relations through which they are connected to the red triangle concept have lower activation weights.

While these capabilities are still far away from the example discussed in the previous chapter, I hope it helps to show how the DPS model can be implemented, and that such a system can in principle handle the required contextual information.

Chapter 7

Discussion of Constraints and Possible Objections

In this chapter I will evaluate the constraints that the DPS model should satisfy. These constraints include those that were described explicitly in chapter 3, but I will also show how the model allows existing contextualist examples to function, and how it deals with some examples that are put forward by other positions. In addition to those I will also discuss a number of potential objections that could be made against the model.

7.1 Limitations and Possible Objections

I will start by discussing some objections that could be made against this model. For some of them I can offer ways in which they could be solved, and for some I do not, but I believe these are not strong enough to disqualify the DPS model as a whole.

7.1.1 Limiting Combinatorial Explosion

The parser as currently described and implemented is a pure bottom up parser that computes every possible interpretation. While this works, it has as a disadvantage that a lot of parse nodes may be generated that have a very low probability value and that are never used for the final interpretation of the entire sentence. Since every combination of patterns is tried the total number of parse nodes may become very large and result in a combinatorial explosion.

The simplest way to handle this is to stop constructing all possible parses of an input. Instead the parser could stop matching on nodes that have a probability that is much lower than other nodes at the same level of the tree structure. In this way the interpretations for the full sentence with the highest probabilities could be found without calculating the entire DAG with alternative interpretations.

The parser could start by always considering the nodes that had the highest probability and try to pattern match on those. If no more matches were found it could move on to nodes with a slightly lower probability value. Newly generated nodes would wait until the probability threshold was lowered enough so that they could be considered. This could continue until a parse was found that provided an interpretation for the entire input. This approach could fail to find the interpretation with the highest probability if it had child nodes with a very low probability, in that case the parser would not attempt to pattern match on these low probability nodes and thus fail to find the interpretation. Such a situation can be prevented if parent nodes always have a lower probability value than the child nodes they are a parent of. In that case a high probability parse of the full input could not have lower probability children. If the probability of a node is always the product of a number of values that are no larger than 1.0 and that include the probability values of its children, this requirement is always satisfied.

The pattern matchers in the example from section 5.4 and in the implemented program in fact meet this requirement, although the program does not implement a parsing strategy as described here.

If there would be a need to use pattern matchers in the parser that do not satisfy this condition, it would need to be considered if it would be a problem if the parser did not always manage to find the most likely parse. After all, human language interpreters are not perfect either.

By using this strategy a combinatorial explosion of parse nodes could be prevented.

7.1.2 Possibility of Communication

A language philosophical question might be how concepts in the brain relate to the external world, and related to this how people are able to communicate with each other if they cannot share "meanings", which are taken to be conceptual structures private to ones mind.

I will argue first of all that there is not necessarily a simple relation between the concepts in ones brain and things in the real world. The conceptual representation of the world that one carries in his mind is formed through the information we get from our senses, and by thought processes. Sometimes we are mistaken about what the world is like, in such a case we have a conceptual representation of a part of the world that does not map directly onto the external world as it actually is. I would argue that since humans have no infallible way of knowing what the world is like, a theory of language must allow for someone to be mistaken about his beliefs about the world while still allowing him to use language successfully. Acquiring an accurate representation of the world is very non trivial, and it is the goal that the entire enterprise of ‘science’ has tasked itself to do. To paraphrase Immanuel Kant, the world as it truly is (the *Ding an sich*) is unknowable, and we can only interpret the world through the categories that our brain bestows upon it.[1]

This view also implies that the notion of ‘reference’ is more problematic than in some other language philosophical models. Concepts may fail to properly refer to anything, and we may not even know it. If we follow this interpretation of Kant, we can *never* be absolutely sure about whether a concept successfully refers to something in the real world.

But the good news is that this unknowability of the world does not prevent communication. All people live in the same world, and all people share the same basic sensory and cognitive capabilities, in other words we have mostly the same hardware. Therefore people normally come up with similar conceptual representations for their surroundings, at least for the more simple parts of it. The similarity in the conceptual graph that two people have in their minds allow them to communicate successfully. While full sharing of thoughts is impossible, if a speaker speaks a thought in the form of a sentence, that sentence can be interpreted by the listener into a conceptual structure that is isomorphic to the conceptual structure that was the speakers original thought. It should be emphasized that this isomorphism is not absolute and only up to a degree, absolutely correct and reliable transmission of thoughts is impossible because conceptual graphs are never exactly the same between two people.[2]

Communication is impossible if the speaker and listener do not share enough of a conceptual structure relevant to the topic of communication.

[1]This paraphrasing is taken far out of the context in which Kant originally used it, so construing it to say that Kant agrees with this interpretation is far fetched. Still, the expression can be re-purposed as a concise formulation of what I want to express. (see [Rohlf, 2010] for what Kant did mean).

[2]As a corollary, this property that it is impossible to convey thoughts exactly but it is possible to convey thoughts mostly accurately makes one think of language meaning as something analog. One might then wonder what it would be like to form a digital equivalent of human language.

Similar to what happens when the switch was made from analog to digital communication technology, one might try to replace the large or nearly infinite set of concepts, which are sometimes hard to differentiate—analogous to the infinite set of symbols that can be used in analog communication—by a very small set of very simple concepts, so that it is easy to recognize which concept was meant. If these concepts then only have a very small number of simple ways in which they can combined, larger structures can be built up from them, and it would still be possible to convey the structure that one has built up in ones mind without error.

This is in fact exactly what happens when we use mathematical language. There are only a small number of basic concepts in math which can only be combined in a small number of ways and which are predictable. From this basis we can then build more complex combinations that we can use to do more advanced things, but it is still possible to convey thoughts about them to other people without loss of fidelity as long as the ideas are transferred in the form of math.

Therefore it may be impossible for a theoretical physicist to explain what he is working on to someone who does not have the necessary background knowledge, unless the physicist takes the time to also explain all the required background knowledge. Doing the latter will allow the listener to extend his conceptual graph so that the original topic of communication can then be discussed.

7.1.3 Complex Language

The examples discussed in this paper thus far all have a very simple sentence structure, usually consisting of only a main clause with a single transitive verb, and only having singular noun phrases. One might wonder how more complex sentences could be handled.

One limitation is that in the current description of the model there is a strict separation between calculation of the activation of concepts and the parsing of input. In reality, if we use sentences longer than a few words it is likely that the first constituents of the sentence influence the probabilities of specific concepts being used in the interpretation of later sentence constituents. Therefore the interpretation of the earlier sentence constituents needs to be able to update the activation values of concepts.

Defining pattern matchers in the parser that can interpret more complex sentences would require a lot of extension of the base model as it is presented in this paper. For plurals the concept graph would probably need to have a way to represent groups of things, and at this point we would probably also require representation of vague relations. It is not always clear which things belong to a group if we use a plural noun phrase, so it could be useful to represent such a group as a probability distribution over the set of objects that could belong to the group. So that would require concept relations in the concept graph that are not binary but are weighted.

Some kinds of generalized quantifiers would probably also require similar probability distributions to represent. For example words like “most”, “all” or “almost all” could be represented as different probability distributions over some scale. How this scale would be applied to a given situation could then be decided contextually. The concept graph would thus need to represent probability distributions as explicit concepts. In this way such words that do not represent objects or actions from the external world can still be interpreted as explicit elements in the concept graph.

Structures such as if-then relations *can* be represented relatively straight forward in this model. The counterfactual situation described by the if-then phrase can be modeled as a different world, similar to how different points in time were modeled in the examples. The same works for fictional stories.

However, more complex language constructions is an area that has not been explored very far yet, and as such it is still a topic for future research.

7.1.4 World Knowledge

Probably the biggest drawback of a model such as the one I present here, is that it requires a very large number of parameters to be tuned, such as the specific structure of the concept graph and the values of disambiguation weights. The DPSP program shows that some success can be had by doing this parameter tuning by hand using trial and error, but this will become more difficult if the concept graph becomes larger. At that point automatic ways of parameter tuning will be needed.

While this is an important problem for building a large scale implementation of this model, it does not prevent this model from being useful in analyzing problems in the philosophy of language, or when thinking about specific problems in linguistics.

But from the point of view of using a model in a computer program, the DPS model is at a disadvantage compared to other models that depend on less contextual information. Models based on the literalist tradition only require a relatively small number of variables to be given a value based on the context, at least compared to the amount of contextual information that must be represented in a DPS concept graph. So for the larger goal of giving a complete description of human language, computer learning techniques will be indispensable in order to find a good concept graph structure and other parameter settings.

7.2 Context Shifting Examples

Subsection 2.3.2 shows a number of examples that are used to make context shifting arguments. These examples, and the argument in general, should continue to work in the DPS model, otherwise it will have failed to be a good contextualist model of language meaning.

7.2.1 The Cat on the Mat

For the "cat on the mat" example we will assume that the orientation of objects is one thing that is represented in the concept graph, with one concept per orientation, and we will also accept that the human mind has a tendency to categorize experiences.

In both situations that Searle describes where cat-mat pairs are floating about in space, i.e. one where they are oriented randomly and one where they are oriented in only two ways, the relation of 'orientation' would get a strong activation because it is highly contrastive for the set of objects under discussion.

In the situation where cat-mat pairs come in only two attitudes and we look at them strapped to our seats in our spaceship, our mind would create a categorization in "cats on mats" and "cats below mats" (or whatever you would call the other category). These categories are, like all other knowledge, encoded as concepts. Since we are seeing cat-mat pairs that fit either of those categories all the time, they would get a high activation value. In this case, we could describe one cat-mat pair with the words “The cat is on the mat”. A listener would be able to correctly interpret this as referring to an instance of the specific category, due to the high activation of the two categories and the high activation of the ‘orientation’ relation. In fact, because of the high contrast in this specific example we could also describe the same situation using several different formulations such as “they are upright” (rather than upside-down) which would still be interpreted as referring to the same category.

In the case where cats and mats are floating about in space in random orientations, such a categorization would not be easily formed. Without some kind of reference orientation it would be difficult to assign a specific orientation as “up” or “down”. Any orientation would receive approximately the same activation level because the cat-mat pairs are seen in all orientations. It would therefore be more difficult to get the interpretation of “cat on the mat” that is available in the other situation.

7.2.2 Cut the Grass

The “cut the grass” example has already been discussed extensively in section 5.4, so I will not discuss it again.

7.2.3 Milk in the Fridge

The “milk in the fridge” example can be treated if we accept that there are different concepts that “milk” could be interpreted as. “Milk” can refer to (non-empty) milk bottles, to the substance ‘milk’, or to bodies of this substance. Probably also to other things, but that is not relevant for now. To correctly predict which of these is chosen we will also need to consider the goal structure again. Presumably concepts that represent certain objects have something like a ‘fit-for’ relation with certain concepts that represent activities or goals. In the context of putting milk in the coffee, the concept representing the activity of putting milk in the coffee, or more generally of consuming milk, would receive a high activation because it is closely related to the concept that represents Hugo’s goal, and therefore the interpretation of “milk” as milk bottles (or other similar milk-containing containers) would also.

For the context of cleaning the fridge we can assume that there is also some goal-like conceptual structure that represents what things are supposed to be like, maybe related to a concept of ‘good’. Cleaning involves removing things that are not supposed to be there, such as milk spills in the fridge. The context of the utterance indicates that something is not the way it is supposed to be after cleaning the fridge, so this activates concepts representing things that are not supposed to be in there and lowers the activation of things that are.

7.3 Recanati's Contextual Relativism Example

The DPS model has no problems at all with the example from subsection 2.2.5 Recanati uses to argue for contextual relativism. In the case where Recanati knows who he is, the word "Recanati" is mapped to the same concept as the word "I" spoken by Recanati, which is the concept that represents himself. In the case where Recanati has forgotten his name, the word "I" is mapped as in the previous case, but the word "Recanati" is no longer associated with that same concept. It is therefore interpreted as a new concept which can have different properties from the "I" concept.

The DPS model does not recognize a form of meaning that corresponds to relativisms unevaluated-utterance form that can still contain certain types of indexicals. According to relativism, interpreting language starts with a sentence which is combined with the context of utterance, which results in some kind of thought form that can still contain indexicals. This last form is then interpreted by the interpreter to yield a form that has a truth value. The DPS model does not recognize this distinction. If anything, the interpreter comes first, and different contexts of utterance can be represented in the interpreters mind as different worlds or as different parts of his concept graph.

7.4 Neurological Realizability

In section 3.3 we discussed some minimal constraints that must be met for a model to be implementable in a computational system such as the human brain. The primary concern regarding neurological realizability is whether a model can be executed on a massively parallel computational platform that has very low sequential performance. For the DPS model, I think it can.

At the parsing stage all recognizers that implement specific production rules can run in parallel, as long as they do not depend on each others outputs. So at least every row of the parse chart can be parsed in parallel. If each parse rule recognizer can run in constant time, the sequential time complexity of parsing an input sentence is proportional to the height of its parse tree, which is on average roughly equivalent to the logarithm of the sentence length. The example sentences that the DPSP program can handle usually do not have a height of more than 5 or 6, so that should not be a problem for the 'at most 30 sequential steps' neurolinguistical constraint. But a more firm conclusion either way will probably require a more detailed model than the simple one I have presented in this paper.

But this requires parse rules to operate in constant time. The main potentially time consuming task for individual parse rules is to find matching concepts. On a Von Neumann type computer every candidate concepts has to be examined individually, so the time complexity is at least linear in the number of concepts examined. On a massively parallel architecture this search can be done in parallel. If we assume each concept in the graph is implemented by its own neurons and each concept can track its own activation, calculation of which concept is the best match can also happen in parallel.

The activation spreading subsystem is not directly constrained by the same constraints as language parsing because it does not need to happen right at the moment that linguistic input is received. The activation algorithm can run in the background, so to say. Having said that, activation can be implemented in a neural network like model very efficiently. The whole idea was developed in the context of connexionist neural networks. The only thing that might be harder to implement in the same way as the DPSP program does is the check to make sure activation does not flow in a loop and amplify itself indefinitely, as this requires all the possible flow paths to be tracked. But with the right tuning of the involved weights maybe problematic amplification of activation levels could prevent even without a loop check.

Since we do not know how the brain works exactly it is hard to list all the requirements that an algorithm that runs on it must satisfy. The above description is therefore a minimal requirement. There is research available that shows how some kinds of processing are implemented in the brain. For example [Werning, 2010], [Werning, 2012] describe how primitive properties such as color and orientation of lines are implemented by having a group of neurons for each position in the visual field for each color or orientation. This requires a lot of neurons, but that appears to be no problem for the brain. [Werning, 2012] gives evidence on how the composition of these primitive properties can happen in the form of eigenmodes of recursive neural networks. An eigenmode of a recursive neural network manifests itself as a repeating temporal firing pattern of the cells that are involved. A recursive neural network can have a lot of eigenmodes. In Wernings interpretation, each eigenmode corresponds to a composition of primitive properties into a complex unit. The primitive properties are represented by individual cells that take part in the pattern. Given certain initial conditions in the network, the temporal patterns for most eigenmodes are not stable and fade out over a short time, while one or a few are amplified. Tentatively applied to the DPS model, such eigenmodes could correspond to complex concepts that represent objects in the environment, and the eigenmode amplification is a mechanism by which an appropriate concept is selected or constructed.

Representation of Continuous Domains

Other interesting research is on the representation of locations in space in the brain[3]. One way in which the brain solves this problem is by having certain cells represent certain points within the space. The cell fires more rapidly if the object of interest is closer to the point it represents. Often, the 'object of interest' is the test animal itself whose neurons are being measured, and the neurons thus represent the location of oneself within a space.

In this case each so-called place cell is not restricted to encoding a single point in a single space somewhere in the world, the whole bundle of place cells is reused for each space that the subject animal is in. So every cell can represent a point in the space the animal currently is in. What point a cell represents in one space appears to be unrelated to the point it represents in a different space, and in fact in each space many cells are not used at all, but which cells differs from space to space.

While this representation of place does not prove a lot regarding how space should be represented in a conceptual graph, it is certainly inspiring. In fact I would not be surprised if other continuous domains would also be represented in a similar way.

7.5 Evolvability

As I discussed in section 3.4, any innate system of the human body or mind must be explainable by evolution somehow. The language system is special in this regard because language is unique to humans. Other systems that are shared with different species have had millions of years to evolve in the predecessors of humans, but language as such has not.

7.5.1 Concept Graph

In the DPS model, a large part of the complexity is taken up by the conceptual graph. The conceptual graph attempts to represent the model of the external world that minds create within them. It is not language specific and therefore not unique to humans, although one could probably argue that humans have a more extensive representation of the world around them than many other species do. It is plausible to assume that when creatures evolved better sensory systems and larger brains, an internal representation of their environment evolved along with that since it is useful for being better able to understand and predict your environment.

[3]This information comes from [Muller, 1996]. Newer information can be found e.g. at Wikipedia and the list of references there: http://en.wikipedia.org/wiki/Place_cell.

7.5.2 Parser

The other part of the DPS model that requires explanation is the mapping from words (or other symbols) to concepts. The production rules as implemented in the DPSP program are actually quite simple, each of them recognizes a specific configuration of features. As such they are fundamentally not very different from for example the line orientation detectors from [Werning, 2012][4], that have probably existed as long as eyes have.

Another useful comparison is with the high level analysis of visual scenes. [Socher et al., 2011] show that a single recursive neural network algorithm can be used to both analyze images and parse sentences, as both have the same kind of recursive structure, where larger units are created from smaller units.

The idea of the 'grandmother neuron' is also related to this. It is the idea that there exist neurons or groups of neurons that respond specifically to certain very complex objects, such as your grandmother. While this idea is not uncontroversial, research has been done on it and [Quian Quiroga et al., 2005] reports on findings of neurons that apparently respond only to images of specific faces or objects, independent of simpler properties such as the image size or the direction from which the object is shown.[5].

While this evidence is not conclusive, it shows that a point can be made that a similar kind of feature detection that is done by parse rules already had a purpose in other systems that existed long before humans evolved, and so that all that needed to happen was for such detectors to be adapted to the auditive system.

7.5.3 Human Specific Evolutions

Given the above, one may wonder what element of the linguistic processing capability humans have is specific to humans. I can not offer much of an answer. One might speculate that it is the specific kind of application of feature detectors to auditory or symbolic inputs, or it could be a more general enlargement of the world representation in the brain, or it could be something else.

7.6 Learnability

As mentioned in section 3.5, I will spend a few words to discuss whether language as described by the DPS model can be considered to be learnable

[4]The line orientation detectors receive input from cells that detect contrast in a specific part of the visual field. That, in turn, is done by receiving inputs from retina cells in specific configurations through excitatory and inhibitory connections.

[5]For more background on the controversy and more research see e.g. http://en.wikipedia.org/wiki/Grandmother_cell.

by children. The language specific parts of the DPS model, i.e. the parser, is relatively easy to learn. Once a concept has formed, all that has to be done is to associate that concept with a word. If then certain patterns in the linguistic input can be recognized, that pattern can be linked to a specific pattern of concepts and relations in the concept graph. Viewed from this angle, learning a language is just an advanced form of pattern recognition.

The weights for the disambiguation methods also should not pose a problem in principle. Many of such weights may simply receive a value proportional to how often the concept or concepts in question are used or used together. If there is not a lot of information available to learn all the required weights, a less than optimally tuned system of weights will produce a poorer performance, but the performance should degrade gradually with more poorly tuned weights. Also, as children have less world knowledge than grownups, we can assume their conceptual graph is smaller, and so the total number of weights is smaller as well.

Learning of the conceptual graph itself is more of a challenge to describe. It requires multiple levels of abstractive capabilities. There are probably all kinds of innate mechanisms involved in this, such as mechanisms for interpreting our two dimensional visual input as a space with objects in it and systems to interpret what other people are thinking and feeling.[6] But regarding how higher levels of abstraction are done I do not have an answer. However this is not a question that is unique to the DPS model. If we agree that the mind carries within it some kind of representation of the world around it, every theory of this will need to explain how one learns it. All I can say is that at a first glance there are no reasons to assume that the model I propose would be much more difficult to learn than other proposals.

[6] [Frith and Frith, 2010], [Hari and Kujala, 2009], [Levinson, 2006], [Garrod and Pickering, 2004]

Chapter 8

Extensions and Future Research

In section 7.1 a number of shortcomings of the model as described here were already mentioned, specifically using a parsing algorithm that does not succumb to a combinatorial explosion, defining parsing strategies and algorithms for parsing more complex sentences, and the problem of world knowledge. In this chapter I will discuss some additional topics that also require more research attention, and also quickly come back to the problem of world knowledge again.

8.1 Concept Graph

As stated earlier, the current design of the concept graph was chosen because it is relatively simple and it is good enough for our purposes at the moment. But I do think that as a model of how the human brain models its surroundigs, it has a limited accuracy.

First of all, the concept graph is rather discrete: either there is a relation between two concepts, or there is not. As connections between neurons in the brain are weighted rather than binary it seems unlikely that the human brain uses this same discreteness in representing relations between things. In the current implementation this issue is mitigated a bit by the activation system, which allows each concept to have a real-valued activation and allows for real-valued relatedness weights, but this is in all likelihood still a crude approximation of how the human brain actually works.

One example where this plays a role is that the current design makes a strict distinction between being an instance of something and having some property. It seems unlikely that real brains make such a strict distinction, and it would be nice if this could be made to work more fluidly in some way.

A complicating factor in improving the concept graph is that the brain most likely does not use one single representation to represent all aspects of its surroundings. Research such as FMRI scanning has shown that different parts of the brain are active when someone is working on different kinds of tasks.[1] Many of these parts have different evolutionary histories and different behaviors, so it seems to be a safe assumption that these different brain areas are also wired differently, and thus represent their data in different ways. I think representing all data in the same way such as the DPS model does can work up to a certain level, but at some point a more accurate model of human thought will be needed to model the actual human brain more closely. How the brain actually does its processing and how it represents the external world in which it lives is ultimately a question for neurologists and psychologists to answer.

Alternatively, if we do not wish to have a fully accurate model of the human brain but just want to have a representation that is simple enough to handle but rich enough to be useful, there are lots of designs possible. Most likely many designs will work better for some linguistic phenomena and other designs for other phenomena. How a conceptual representation should be structured to handle most or all linguistic phenomena is something that will require a lot more research and experimenting.

8.2 Accessibility of Concepts

It is obvious that in real life, in different situations and contexts some concepts are more easily accessible than others. This is why we do not usually need to think consciously about which interpretation was meant when an ambiguous word is used. The DPS model tries to model this by giving every concept in the concept graph an activation level, and describes some mechanisms by which the activation is updated. In the real brain there are probably other factors involved as well that influence which concepts are accessible.

One such factor that is not modeled is attention. Attention is certainly influenced by our environment, but we can also voluntarily choose to focus our attention on specific things.[2] Attention is drawn e.g. by loud sounds or flashes of light, or by unexpected movement. Ideally an implementation of activation should also take such factors into account, but especially voluntary control of attention will probably be beyond the reach of science for some time to come.

Associations between concepts do not need to be based on anything intrinsic to the concepts, they can also be created explicitly. The memory technique of loci for remembering a list of items is based on this:[3] it is easier to remember a list of items by associating each item with for example

[1] [Huettel et al., 2009]
[2] [Theeuwes, 1991]
[3] [Carlson, 2010, p. 245]

a thing in your house. You can then recall the list by mentally walking through your house past all the things you associated something with. A similar phenomenon is that some people who study while listening to music then associate the music they listened to to the subject they studied.

Ideally the method to compute the accessibility of concepts should be able to take all such mechanisms into account. Unfortunately that may be difficult to implement just like that. I hope to show with this model and program that even without implementing all these mechanisms they can provide some insight in thinking about the semantics of language. The more of these mechanisms can be identified and described the more useful an activation system should become for identifying correct interpretations of language.

8.3 Spoken Language Interpretation

Many current automatic speech recognition systems are based on hidden markov models[4]. One property of such systems is that they are inherently based on probabilities. If the DSP parsing mechanism were adapted to make predictions on the next expected word, that would open up possibilities to send information on probabilities all the way from the semantic layer to the lowest speech recognition layers. Since in general for probabilistic or statistic applications more data means better results,[5] this would potentially allow improved speech recognition systems to be developed.

The parser as described by the DPS model could be made probabilistic in relatively straightforward ways. The first step would be to make it parse input on-line, as each word comes in. This is a relatively straightforward change as the different pattern matchers that make up the parser operate independently. The next change would be to see which pattern matchers could match when the next input word becomes available given the current state of the partially built parse tree. Based on this a set of words can be selected that allow at least one recognizer to match. These words could be given a probability based on the probability values that pattern matchers would give to the parse nodes they would create if one of such words was given as input. Depending on the exact implementation of the pattern matchers such a probability might be more or less simple to calculate.

8.4 Language Production

The current description of DPS and the DPSP program only handle language understanding, not language production. I think it should be possible to extend the system to work ‘in reverse’, where certain concepts and relations

[4]See [Jurafsky and Martin, 2008], specifically chapter 9

[5] [Halevy et al., 2009]

between them are transformed into symbolic output. However that is also a topic that will probably require some experimentation to get to work, so it is a topic for future research.

8.5 World Knowledge

On the practical side and as noted in subsection 7.1.4 before, creating an actually useful program along the lines of DPSP quickly runs into the problem of how to create a large conceptual graph that contains a rich representation of the world, and how to tune the large number of activation weights that come with such a graph. Formalizing large amounts of world knowledge is not something that is easy to do. It will probably only become feasible if we find some way to have the computers learn the required information for themselves.

However in the medium term it might be possible to program knowledge about for example 2d or 3d spaces and objects into a concept graph, and have a program talk about those things in a relatively natural way. Perhaps building a starting set of information plus a way to extend the concept graph (though both would be nontrivial tasks in their own right) could be a beginning from which a program could start to learn more information about the world around it.

Chapter 9

Conclusion

In conclusion, we can say that it is possible to formalize a model of linguistic meaning within the contextualist school of thought. The model that was presented is named Direct Probabilistic Semantics. It can give interpretations for several examples that have been used in the literature in the debate of literalism versus contextualism, which agree with the way many contextualists would want them to be interpreted. In addition, the model, as far as it has been developed, seems to fit within the constraints that any computational model that aims to describe how something is done in the human brain must fit.

In addition, to prove that the model can be made fully precise, and to aid in experimentation, I have written a proof of concept implementation in a computer program.

Both the model and the implementation written still miss a lot of desirable properties, but as of yet the theoretical problems that were considered did not appear to be insurmountable. On the practical side there is still the problem of getting a large amount of world knowledge into a formal system, but this problem is shared with all other existing theories of language.

Hopefully this result will help clarify the discussion between literalists and contextualists. We can also hope this result will help the participants in the discussion to agree on a shared point of view and thus end the discussion, but with discussions in the field of philosophy, such hopes of ending a discussion are usually idle. Regarding the field of linguistics and computational linguistics, this result may help promote the acceptance of approaches that take more contextual information into account.

Bibliography

[Anderson, 1983] Anderson, John R., “A spreading activation theory of memory”, *Journal of Verbal Learning and Verbal Behavior*, vol. 22: pp. 261–295 (1983).

[Asher and Pustejovsky, 2005] Asher, Nicolas and James Pustejovsky, “Word Meaning and Commonsense Metaphysics”, (2005), in course materials for Type Selection and the Semantics of Local Context, ESSLLI 2005, available from http://semanticsarchive.net/Archive/TgxMDNkM/.

[Aswath et al., 2005] Aswath, Dipti, Syed Toufeeq Ahmed, James D’cunha, and Hasan Davulcu, “Boosting Item Keyword Search with Spreading Activation”, in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, WI ’05, pp. 704–707, IEEE Computer Society, Washington, DC, USA (2005), ISBN 0-7695-2415-X, doi:10.1109/WI.2005.44.

[Austin, 1961] Austin, J., *Philosophical Papers*, Oxford University Press (1961), 3rd ed. (1979).

[Bertolo, 2001] Bertolo, S., ed., *Language Acquisition and Learnability*, Cambridge University Press (2001).

[Bianchi, 2010] Bianchi, C, “Contextualism”, in Horn, Laurence and Gergory Ward, eds., *Handbook of Pragmatics*, John Benjamins Publishing Company (2010).

[Borg, 2004] Borg, Emma, *Minimal Semantics*, Oxford University Press (2004).

[Bos et al., 2003] Bos, Johan, Ewan Klein, Oliver Lemon, and Tetsushi Oka, “DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture”, in *In 4th SIGdial Workshop on Discourse and Dialogue*, pp. 115–124 (2003).

[Cappelen and Lepore, 2005] Cappelen, Herman and Ernest Lepore, *Insensitive Semantics. A Defense of Semantic Minimalism and Speech Act Pluralism*, Blackwell Publishers (2005), ISBN 9781405126755.

[Cappellen, 2006a] Cappellen, Herman, "Reply to Critics", *Philosophy and Phenomenological Research* (2006a).

[Cappellen, 2006b] Cappellen, Herman, "Reply to Critics", *ProtoSociology* (2006b).

[Carlson, 2010] Carlson, Neil R., "Psychology the science of behaviour", *Pearson Canada Inc* (2010).

[Carston, 2002] Carston, R., *Thoughts and utterances: the pragmatics of explicit communication*, Blackwell (2002).

[Charniak, 1972] Charniak, E., "Toward A Model Of Children's Story Comprehension", Tech. Rep. AI-TR-266, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (1972), available at http://dspace.mit.edu/bitstream/handle/1721.1/6892/AITR-266.pdf?sequence=2.

[Chomsky, 1986] Chomsky, N., *Knowledge of language: Its nature, origins, and use*, Greenwood Publishing Group (1986).

[Chomsky, 2000] Chomsky, N., *New Horizons in the Study of Language and Mind*, Cambridge University Press (2000).

[Cohen, 1986] Cohen, Jonathan, "How is Conceptual Innovation Possible?", *Erkenntnis*, vol. 25: pp. 221–238 (1986).

[Collins and Loftus, 1975] Collins, Allan M. and Elizabeth F. Loftus, "A spreading-activation theory of semantic processing", *Psychological Review*, vol. 82, no. 6: pp. 407–428 (1975).

[Coon, 1989] Coon, Dennis, *Introduction to Psychology, Exploration and Application*, West Publishing Company, St. Paul (1989).

[Davidson, 1984] Davidson, Donald, *Inquiries into Truth and Interpretation*, Clarendon Press (1984).

[Doya et al., 2007] Doya, Kenji, Shin Ishii, Alexandre Pouget, and Rajesh P.N. Rao, eds., *Bayesian Brain: Probabilistic Approaches to Neural Coding*, The MIT Press (2007), ISBN 9780262042383.

[Fletcher and MacWhinney, 1995] Fletcher, P. and B. MacWhinney, *The Handbook of Child Language*, Blackwell, Cambridge, MA (1995).

[Frith and Frith, 2010] Frith, U. and C. Frith, "The social brain: allowing humans to boldly go where no other species has been", *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 365, no. 1537: pp. 165–176 (2010).

[Garrod and Pickering, 2004] Garrod, S and M.J. Pickering, “Why is conversation so easy?”, *Trends in Cognitive Sciences*, vol. 8: pp. 8–11 (2004).

[Geurts and Beaver, 2011] Geurts, Bart and David I. Beaver, “Discourse Representation Theory”, in Zalta, Edward N., ed., *The Stanford Encyclopedia of Philosophy*, fall 2011 edn. (2011), http://plato.stanford.edu/archives/fall2011/entries/discourse-representation-theory/.

[Grice, 1989] Grice, H.P., *Studies in the Way of Words*, Harvard University Press (1989), ISBN 9780674852716.

[Halevy et al., 2009] Halevy, Alon, Peter Norvig, and Fernando Pereira, “The Unreasonable Effectiveness of Data”, *IEEE Intelligent Systems*, vol. 24, no. 2: pp. 8–12 (2009), ISSN 1541-1672, doi:10.1109/MIS.2009.36.

[Hari and Kujala, 2009] Hari, R. and M. V. Kujala, “Brain basis of human social interaction: from concepts to brain imaging”, *Physiological reviews*, vol. 89, no. 2: pp. 453–479 (2009).

[Heim, 1982] Heim, I., “The Semantics of Definite and Indefinite Noun Phrases”, Ph.D. thesis, University of Massachusetts, Amherst (1982).

[Huettel et al., 2009] Huettel, S. A., Song A. W., and G. McCarthy, *Functional Magnetic Resonance Imaging*, Sinauer, Massachusetts, second edn. (2009), ISBN 978-0-87893-286-3.

[Jurafsky and Martin, 2008] Jurafsky, Daniel and James H. Martin, *Speech and Language Processing*, Pearson Prentice Hall, second edn. (2008), ISBN 978-0131873216.

[Kamp, 1981] Kamp, Hans, “A theory of truth and semantic representation”, in Groenendijk, J. A. G., T. M. V. Janssen, and M. B. J. Stokhof, eds., *Formal Methods in the Study of Language*, pp. 277 – 322, Mathematical Centre Tracts, Amsterdam (1981).

[Kanis, 2011] Kanis, Jan, “For and Against Objective Meaning”, Bachelor thesis, Universiteit Utrecht (2011).

[Kripke, 1976] Kripke, Saul, “Is there a Problem about Substitutional Quantfication?”, in Evans, Gareth and John McDowell, eds., *Truth and Meaning*, Oxford University Press (1976), ISBN 978-0-19-825007-4.

[Levinson, 2006] Levinson, S.C., “Cognition at the heart of human interaction”, *Discourse Studies*, vol. 8, no. 1: pp. 85–93 (2006).

[MacFarlane, 2007] MacFarlane, J., "Semantic Minimalism and Nonindexical Contextualism", in Preyer, G. and G. Peter, eds., *Context-Sensitivity and Semantic Minimalism. New Essays on Semantics and Pragmatics*, pp. 240–250, Oxford University Press (2007).

[MacFarlane, 2009] MacFarlane, J., "Nonindexical Contextualism", *Synthese*, vol. 66: pp. 231–250 (2009).

[Madsen, 2009] Madsen, Mathias Winter, "The Limits of Machine Translation", Master thesis, University of Copenhagen (2009).

[Manning and Schütze, 1999] Manning, Christopher D. and Hinrich Schütze, *Foundations of Statistical natural Language Processing*, MIT Press (1999), ISBN 978-0-262-13360-9.

[Meltzoff, 2002] Meltzoff, A. N., "Imitation as a mechanism of social cognition: Origins of empathy, theory of mind, and the representation of action", in Goswami, U., ed., *Handbook of childhood cognitive development*, pp. 6–25, Blackwell Publishers, Oxford (2002).

[Montague, 1970a] Montague, Richard, "English as a Formal Language", in et al., Bruno Visentini, ed., *Linguaggi nella Società e nella Tecnica*, pp. 189–224, Edizioni di Comunità, Milan (1970a), reprinted in Montague 1974, 188-221.

[Montague, 1970b] Montague, Richard, "Universal Grammar", *Theoria*, vol. 36: pp. 373–398 (1970b), reprinted in Montague 1974, 222-246.

[Montague, 1973] Montague, Richard, "The proper treatment of quantification in ordinary English", in Hintikka, K.J.J., J.M.E. Moravcsik, and P. Suppes, eds., *Approaches to Natural Language*, pp. 221–242, Reidel, Dordrecht (1973), reprinted in Montague 1974, 247-270; Reprinted in Portner and Partee, eds., 2002, 17-34.

[Montague, 1974] Montague, Richard, *Formal Philosophy. Selected Papers of Richard Montague*, Yale University Press, New Haven/London (1974), edited and with an introduction by Richmond H. Thomason.

[Muller, 1996] Muller, Robert, "A Quarter Century of Place Cells", *Neuron*, vol. 17: pp. 979–990 (1996).

[Och et al., 1999] Och, F. J., C. Tillmann, and H. Ney, "Improved alignment models for statistical machine translation", in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 20–28 (1999).

[Parker, March 2006] Parker, Anna R., "Evolution as a Constraint on Theories of Syntax: The Case against Minimalism", Ph.D. thesis, University of Edinburgh (March 2006).

[Partee, 2008] Partee, Barbara H., “Reflections of a formal semanticist”, in Partee, Barbara H., ed., *Compositionality in Formal Semantics: Selected Papers of Barbara H. Partee*, John Wiley & Sons (2008).

[Pietrosky, 2003] Pietrosky, Paul M., “The Character of Natural Language Semantics”, in Barber, Alex, ed., *Epistemology of Language*, Oxford University Press (2003), available at http://www.terpconnect.umd.edu/~pietro/research/papers/index.html.

[Pietrosky, 2005] Pietrosky, Paul M., “Meaning Before Truth”, in Preyer, G. and G. Peters, eds., *Contextualism in Philosophy*, Oxford University Press (2005), available at http://www.terpconnect.umd.edu/~pietro/research/papers/index.html.

[Quian Quiroga et al., 2005] Quian Quiroga, R., L. Reddy, G. Kreiman, C. Koch, and I. Fried, “Invariant visual representation by single neurons in the human brain”, *Nature*, vol. 435: pp. 1102–1107 (2005), doi:10.1038/nature03687.

[Recanati, 1993] Recanati, F., *Direct Reference: From Language to Thought*, Blackwell (1993).

[Recanati, 2001] Recanati, François, “What is Said”, *Synthese*, vol. 128: p. 75–91 (2001).

[Recanati, 2004] Recanati, François, *Literal Meaning*, Cambridge University Press (2004), ISBN 9780521537360.

[Recanati, 2007] Recanati, François, *Perspectival Thought: A Plea for (Moderate) Relativism*, Oxford University Press (2007), ISBN 9780199230549.

[Recanati, 2008] Recanati, F., “Pragmatics and Semantics”, in *The Handbook of Pragmatics*, Blackwell Publishing Ltd, Oxford, L. R Horn and G. Ward (2008), doi:10.1002/9780470756959.ch20.

[Reddy, 1979] Reddy, Michael J., “The conduit metaphor: A case of frame conflict in our language about language”, in Ortony, Andrew, ed., *Metaphor and Thought*, pp. 284–310, Cambridge University Press (1979), second edition: 1993.

[Reisberg, 2007] Reisberg, Daniel, *Cognition: Exploring the Science of the Mind*, W W Norton & Co. Inc. (2007).

[Ritchie and Bhatia, 1999] Ritchie, W. and T. Bhatia, *Handbook of Child Language Acquisition*, Academic Press, San Diego (1999).

[Rohlf, 2010] Rohlf, Michael, “Immanuel Kant”, in Zalta, Edward N., ed., *The Stanford Encyclopedia of Philosophy*, fall 2010 edn. (2010), http://plato.stanford.edu/archives/fall2010/entries/kant/.

[Samsom, 2013] Samsom, Dana, “Theory of Mind”, in Reisberg, Daniel, ed., *The Oxford Handbook of Cognitive Psychology*, Oxford University Press (2013).

[Searle, 1978] Searle, John R., “Literal Meaning”, *Erkenntnis*, vol. 13, no. 1: pp. 207–224 (1978).

[Searle, 1979] Searle, John R., *Expression and Meaning*, Cambridge University Press (1979).

[Searle, 1980] Searle, John R., “The Background of Meaning”, in Searle, John R., Ference Kiefer, and Manfred Bierwisch, eds., *Speech Act theory and Pragmatics*, pp. 221–232, Reidel (1980).

[Searle, 1992] Searle, John R., *The Rediscovery of Mind*, MIT Press (1992).

[Socher et al., 2011] Socher, Richard, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning, “Parsing Natural Scenes and Natural Language with Recursive Neural Networks”, in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA (2011).

[Speaks, 2011] Speaks, Jeff, “Theories of Meaning”, in Zalta, Edward N., ed., *The Stanford Encyclopedia of Philosophy*, summer 2011 edn. (2011), http://plato.stanford.edu/archives/sum2011/entries/meaning/#ChoIntSem, section 2.2.2: Chomskyan Internalist Semantics.

[Stanley and Szabò, 2000] Stanley, Jason and Zoltan Szabò, “On Quantifier Domain Restrictions”, *Mind and Language*, vol. 15, no. 2: pp. 219–261 (2000).

[Stanley, 2000] Stanley, Jason, “Context and Logical Form”, *Linguistics and Philosophy*, vol. 23: pp. 391–434 (2000).

[Theeuwes, 1991] Theeuwes, J., “Exogenous and endogenous control of attention — the effect of visual onsets and offsets”, *Perception & Psychophysics*, vol. 49, no. 1: pp. 83–90 (1991).

[Travis, 1989] Travis, Charles, *The Use of Sense: Wittgenstein’s Philosophy of Language*, Oxford University Press (1989).

[Travis, 1997] Travis, Charles, “Pragmatics”, in Hale, B. and C. Wright, eds., *A Companion to the Philosophy of Language*, p. 87–107, Blackwell (1997).

[Travis, 2000] Travis, Charles, *Unshadowed Thoughts*, Harvard University Press (2000).

[Travis, 2008] Travis, Charles, *Occasion Sensitivity: selected essays*, Oxford University Press (2008), ISBN 978-0-19-923033-4.

[Tversky, 1997] Tversky, “Features of Similarity”, *Psychological Review*, vol. 84: p. 327–52 (1997).

[Werning, 2010] Werning, Markus, “Complex First? On the Evolutionary and Developmental Priority of Semantically Thick Words”, *Philosophy of Science*, vol. 77, no. 5: pp. 1096–1108 (2010).

[Werning, 2012] Werning, Markus, “Non-Symbolic Compositional Representation and its Neural Foundation: towards and Emulative Semantics”, in Werning, Markus, Wolfram Hinzen, and Edouard Machery, eds., *The Oxford Handbook of Compositionality*, pp. 633–724, Oxford University Press (2012).

[Wikipedia, a] Wikipedia, “http://en.wikipedia.org/wiki/Conduit_metaphor”, (a), accessed August 2013.

[Wikipedia, b] Wikipedia, “http://en.wikipedia.org/wiki/Frederick_Jelinek”, (b), accessed August 2013.

[Wikipedia, c] Wikipedia, “http://en.wikipedia.org/wiki/Grandmother_cell”, (c), accessed August 2013.

[Wikipedia, d] Wikipedia, “http://en.wikipedia.org/wiki/Place_cell”, (d), accessed July 2013.

[Wikipedia, e] Wikipedia, “http://en.wikipedia.org/wiki/Spreading_activation”, (e), accessed July 2013.

[Winograd, 1971] Winograd, Terry, “Procedures as a Representation for Data in a Computer Program for Understanding Natural Language”, Tech. Rep. AI-TR-235, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts (1971), available at http://dspace.mit.edu/bitstream/handle/1721.1/7095/AITR-235.pdf?sequence=2.

Appendix A

Program Capabilities

This appendix contains descriptions of some of the capabilities of the DPSP program not mentioned before. The information from the program's README file is also included.

A.1 Locations and Demonstratives

The scene that the concept graph represents is a two-dimensional space, with each object having a location in that space. The program implements the demonstratives "die", "dat" and "daar" ("that" male/female, "that" neutral, "there"), that can be used to refer to locations in the scene. To represent the space itself the concept graph contains a set of instances of 'location', which are arranged in a 200 by 200 grid. Each such location concept thus represents a point in the space. Objects are at a certain location if they have the 'isat' relation with the concept that represents the corresponding coordinate.

For the interpretation of demonstratives, the program asks the user to click somewhere on the scene, which is interpreted as pointing to that location. The demonstratives "die" and "dat" together with such a pointing action are interpreted as different things: first of all a set of possible locations, with the ones closer to the point where the click happened having a higher probability, secondly as any objects at one of those locations, and third as any of the properties any objects at that location have. So saying "die kleur" ("that color") while pointing at a green object will be interpreted as the concept 'green'. The demonstrative "daar" is interpreted similarly but only as locations, not as objects at those locations.

The interpretation of demonstratives is especially a case in which having the conceptual world representation available during parsing results in a much simpler representation compared to approaches in which syntactic or semantic analysis happens without the availability of such a context, because indexicals do not need to be propagated through layers of context independent representations but can be resolved immediately.

A.2 Commands and Statements

The program can not only recognize noun phrases as demonstrated in Table 6.1. It also recognizes some imperative and some predicative statements. The imperative statements allow one to create or move objects or to change their color, using sentences such as “Maak dat vierkant rood” (“Make that square red”), or “Maak een cirkel” (“Make a circle”). These statements are implemented by creating temporary concepts that represent the action to be taken. Temporary concepts do not differ from normal concepts in any way, except that they are removed again when the program is done with the parse and the resulting concept.

The execution of the created statement is not something that happens automatically in the parser subsystem. It is initiated by the driver code in the graphical interface subsystem. Since the concept representing the action already has all the necessary information, execution is straight forward to implement in a program such as this one. As it is not part of how language works it is not really relevant to discuss here in detail.

The program also interprets simple predicative statements, such as “de cirkel is rood” (“the circle is red”). The result is a temporary concept that represents the statement, which can be evaluated in a straightforward way. The graphical interface will evaluate predicative statements and display the result along with the interpretation. I have not implemented any kind of complex predicates or logical operators. Doing so would be possible but I did not have time for it and it was not relevant for my research.

A.3 Limitations on Continuous Values

The program currently can only handle discrete concepts. This shows up in for example there only being a small number of colors available and the space of the scene being represented by a large number of discrete concepts, each of which represents a point on the grid. The latter already causes a notable slowdown in parsing when locations are involved. In a neural network type of underlying architecture, rather than a Von-Neumann type computer, such concerns could be lessened quite a bit as in many cases concepts could be processed in parallel. However that does not solve everything. For domains that contain continuous values and that are unbounded, or that need to be represented at arbitrary resolution, an infinite number of concepts would be needed if they are to be represented in the same way as the implementation currently represents space, so that is obviously untenable.

For the problem of computational efficiency I can think of two ways to improve the current program. The first would be to use the computer hardware more efficiently. A space could be represented by an efficient matrix instead of by a set of objects as is currently the case. Using built-in matrices in many programming languages gives access to operations that are implemented using specific processor instructions for operating on matrices, which can process multiple values in parallel. This would therefore give access to a certain level of parallel processing, and reduce the disadvantage a Von Neumann architecture has over a neural network.

The second way would be to use a more analytic approach, in which instances of 'location' are created when needed. Activation of a point and the area around it could be represented by a set of gaussians, in which case more use could be made of analythic methods to calculate the most probable interpretation of a set of possible interpretations for an input sentence. In such an approach an infinite number of concepts could be represented implicitly, while only those that are needed individually need to be explicit.

A.4 Program Documentation

The rest of this appendix contains a copy of the information that is included in the programs README file. This includes instructions on how to use the program and a short overview of how it is structured.

About

This program is a proof of concept implementation of a contextualist semantic parsing model. This program was written by Jan Kanis as part of my master thesis in Cognitive Artificial Intelligence.

The source code for this program will be available at `http://bitbucket.org/JanKanis/DPSP`. If it is not there you can try to find it through my personal webpage at `http://www.jankanis.nl`.

If you find this program or the thesis it is part of helpful or useful, I would appreciate it if you could let me know at jan dot code at jankanis dot nl.

Name

This program is called "Direct Probabilistic Semantic Parser", or DPSP for short. The name is based on the name of the implemented theoretic approach, which I have named "Direct Probabilistic Semantics".

Theory

The goal of this program is to implement a natural language semantics system based on the contextualist approach, as it is known in the field of philosophy of language. The primary goals are to interpret language phrases directly into concepts, without going through symbolic intermediate representations based on formal logic like many computational semantic approaches do. These symbolic intermediate representations normally cannot represent the rich context that language interpretation depends on.

This program is a proof of concept, its goal is to show that a computational semantic system according to the ideas of contextualism is possible, and does not need to run into inconsistencies or hidden problems.

For more information see my master thesis.

Dependencies

This program is written in *Python 3*, so you will need that installed. You will also need the *numpy* library. If you want to use the graphical interface you will need the *Qt 4 libraries* and *PyQt4*. The tests are written against the *py.test* framework.

The *numpy* dependency is not very important but was convenient. You can remove it by changing the world grid in `dpsp.py` and the functions that use in in `parser.py`.

This program has been developed and tested on Ubuntu Linux 12.04.

Usage

To run the program, start the `gui.py` file. This will open up a window containing three areas. The left half of the window displays the scene. The right lower area is the input area, and the right upper area displays the output.

You can enter sentences into the input area by typing there. Sentences can only consist of the words the program understands, separated by whitespace. The program currently does not understand punctuation or capital letters, if you use them the parse will fail.

Although the interface and documentation of this program is in English, the language the program understands is Dutch.

If you enter a sentence into the input area, the program will attempt to parse the input and show any interpretations it can come up with. Inputs need not be fully formed sentences, noun phrases are also acceptable. The program knows about circles, squares, and triangles, and about the colors red, yellow, green and blue. The program understands commands to make new objects or to change objects. For the full list of words the program understands, see the Lexicon section.

Quick Tutorial

When the program starts, the scene contains two objects: a green square in the left upper quadrant and a red triangle near the center. Input "`het vierkant`" ("the square"). The output window will display the input and a representation of the interpretation, which in this case is

```
<vk1 (80039)>: 0.1984
```

This indicates that the result is the concept that carries the name "vk1" and has id number 40039. The 0.1984 is the probability of this result. This number is not very relevant by itself, only its relative magnitude compared to other interpretations matters.

Other phrases such as "`het groene ding`" or "`de driehoek`" ("the green thing", "the triangle") result in the same square and the triangle concept, respectively. The phrase "`het ding`" ("the thing") is ambiguous as it can refer to the square or the triangle. Inputting it will display both results, with the one with the highest probability first.

The program also supports demonstratives. If you input "`dat ding`" ("that thing") the program will ask you to click on the scene to indicate where you intend "dat" to point to. The demonstrative word in question will be highlighted and the status bar will ask you to click on the scene. If you do so the result will be the object you clicked at, or if you did not click near an object there will be no interpretation available. Clicking is fuzzy, so clicking near an object instead of on it will also work.

You can change locations or properties of objects by issuing sentences such as "`maak het vierkant blauw`" ("make the square blue"). "`zet de driehoek daar`" ("put the triangle there") also works, after you click on a location. This last sentence will result in a lot of possible interpretations due to the fuzziness of demonstratives, all points near the point you clicked will be interpreted as possible targets, but the one you clicked on will have the highest probability.

New objects can be created using sentences such as "`maak een gele cirkel`" ("make a yellow circle") or "`zet daar een vierkant neer`"[*] ("put down a square over there").

In the case of ambiguity, the program will try to choose the most likely interpretation. This allows for example the following sentences to work as expected if they are input one after another (if there is at least a red and a green square): "`zet het groene vierkant daar`" "`zet de rode daar`" ("put the green square there", "put the red one there").

[*]There is currently still a bug that prevents this specific example from working.

Lexicon

The program can understand the following Dutch words:

Dutch	**English translation**
de	the (m/f)
het	the (n)
een	a
die	that (n)
dat	that (m/f)
daar	there
hem	him/it
vierkant	square
driehoek	triangle
cirkel	circle
ding	thing
kleur	color
plek	place
rood	red
rode	red (adj)
geel	yellow
gele	yellow (adj)
groen	green
groene	green (adj)
blauw	blue
blauwe	blue (adj)
op	on
maak	make
zet	put
neer	down
is	is

Some example phrases:

“het vierkant”	“the square”
“het ding”	“the thing”
“de driehoek”	“the triangle”
“maak een gele cirkel”	“make a yellow circle”

“zet het vierkant daar”	“put the square there”
“maak dat ding blauw”	“make that thing blue”
“die cirkel is geel”	“that circle is yellow”

For a full list of recognized concepts, relations, and for the implemented language grammar rules you will need to consult the source. `dpsp.py` contains the concepts and relations, `parser.py` the grammar rules.

Program Layout

The program consists of a number of files:

```
dpsp.py
dpsp_activation.py
dpsp_actions.py
dpsp_utils.py
parser.py
parser_algorithm.py
dpsp_views.py
guicall.py
gui.py
namespace.py
QtAutoconnect.py
```

Additionally there are a number of files containing tests:

```
test_dpsp.py
test_parser.py
test_guicall.py
test_gui.py
test_memoizer.py
test_QtAutoconnect.py
test
```

There are also some documentation and licensing files.

The files `dpsp.py`, `dpsp_activation.py`, `dpsp_actions.py`, `dpsp_utils.py`, `parser.py` and `parser_algorithm.py` comprise the core of the program. `dpsp.py` contains the classes and functions to manipulate the concept graph and the core concept and relation definitions. `dpsp_activation.py` contains the logic to maintain the concept activation. `dpsp_actions.py` contains the functions that execute imperatives and evaluate propositions. `dpsp_utils.py` contains some utilities that are needed in multiple modules. The parser is contained in `parser.py` and `parser_algorithm.py` as the name suggests. `parser_algorithm.py` contains the implementation of the parser algorithm, while `parser.py` contains the languages production rules. The latter also contains some tests of the parser, these are not part of `test_-parser.py` because they depend on the state of the concept graph to be unmodified from the program startup default.

The files `dpsp_views.py`, `guicall.py` and `gui.py` comprise the graphical interface. `dpsp_views.py` contains the definition for the Qt widget that displays the scene and updates it automatically if the concept graph changes. `gui.py` contains the main gui window and the logic to drive it, calling into the parser code as needed. `guicall.py` contains code to make safe calls into the gui from non-gui threads.

The files starting with `test_` contain tests for their respective modules. All tests can be run by executing the `test` script. They are run by the `py.test` testing framework.

The graphical interface can be run as a multithreaded program, in which case parsing of sentences happens on a background thread. This is enabled by setting the variable `ENABLE_THREADED_PARSING` in `gui.py` to `True`.

The gui is decoupled from the parser and concept graph and uses an observer pattern to receive graph updates. It is also possible to use the concept graph and the parser independently of the graphical interface. They don't have their own interface, but it is possible to interact with them through a python interactive interpreter. The main concept graph consists of the set of concepts that are contained in `dpsp.Cl` and `dpsp.C` (the former is indexed by concept id, the latter by name). The relations exist in the set `dpsp.relations`, however separate indexes are attached to each concept so manipulating them needs to be done through the provided functions in `dpsp`.

For programmatic or interactive interpreter interaction with the parser, the main function is `parser.parse([list-of-words])`. It takes a list of strings as argument and returns a `parser.Parse` object that contains all information related to the parse.

License

This program is copyrighted by Jan Kanis © 2013.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version, with the additional condition that you must preserve author attributions. See the `LICENSE.txt` file for details.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Appendix B

DPSP Parser Productions

This appendix contains the full parser definition as it is used in the DPSP program. This code is taken directly from the `parser.py` source file.

```
@parser({'word':'het'}, name="LidwoordRec('het')")
@parser({'word':'de'}, name="LidwoordRec('de')")
def LidwoordRec(pcontext):
  f = pcontext.match[0].features
  gender = {'de':'mv', 'het':'o'}[f['word']]
  yield SyntaxMeaning("LidWoord('{}')".format(f['word']), pcontext
      , dict(f, determiner=True, definite=True, gender=gender,
      word=None))

@parser({'word':'een'}, name="LidwoordRec('een')")
def IndetLidwoordRec(pcontext):
  f = pcontext.match[0].features
  yield SyntaxMeaning("LidWoord('{}')".format(f['word']), pcontext
      , dict(f, determiner=True, definite=False, word=None))


@parser({'word': 'die'})
@parser({'word': 'dat'})
def IndexicalRec(pcontext):
  f = pcontext.match[0].features
  word = f['word']
  gender = {'die':'mv', 'dat':'o'}[word]
  yield IndexicalSyntaxMeaning("Indexical('{}')".format(word),
      pcontext,
      dict(f, determiner=True, definite=True, indexical=True,
          gender=gender, word=None))

@parser({'word':'die'})
def DirectIndexicalRec(pcontext):
  for con, p in pcontext.match[0].getreferents().items():
    # type = con.isinstance
    if C.object in con.types():
      yield con, {}, prob_or(p, activation[p])

```

```

@parser({'word': 'daar'})
def DaarRec(pcontext):
  for con, p in pcontext.match[0].getreferents().items():
    if con.testrel(C.instanceof, C.location):
      yield con, {'prepphrase':True, 'preptype':location}, prob_or
          (p, activation[p])

@parser({'word':'hem'})
def HemRec(pcontext):
  for con in C.object.fullextension():
    yield con, {}, activation[con]


@parser({'word': 'vierkant'})
def SquareRec(pcontext):
  yield C.square, {'gender':'o'}

@parser({'word': 'driehoek'})
def TriangleRec(pcontext):
  yield C.triangle, {'gender':'mv'}

@parser({'word': 'cirkel'})
def CircleRec(pcontext):
  yield C.circle, {'gender':'mv'}

@parser({'word': 'ding'})
def DingRec(pcontext):
  yield C.object, {'gender': 'o'}

@parser({'word':'kleur'})
def KleurRec(pcontext):
  yield C.color, {'gender': 'mv'}

@parser({'word': 'plek'})
def PlekRec(pcontext):
  yield C.location, {'gender': 'mv'}


for w, con in{'rood rode': C.red, 'geel gele': C.yellow, 'groen
    groene': C.green, 'blauw blauwe': C.blue}.items():
  w1, w2 = w.split()
  @parser({'word':w1})
  def ColorRec(pcontext, con=con, w1=w1):
    yield con, {'determined':True}
    yield pcontext.transientconcept("color('{}')".format(w1), {
        istype:indeterminate, iscolor:con}), dict(pcontext.match
        [0].features, adjectival=True, word=None)
  @parser({'word':w2})
  def ColorRec(pcontext, con=con, w2=w2):
    yield pcontext.transientconcept("color('{}')".format(w2), {
        istype:indeterminate, iscolor:con}), dict(pcontext.match
        [0].features, adjectival=True, word=None)

```

```

@parser({'word':'op'}, {instanceof_t: location})
def OpPhraseRec(pcontext):
  yield pcontext.match[1].concept, {'prepphrase':True, 'preptype':location}



@parser({'word':'maak'}, {instanceof_t: obj, istype:None}, {instanceof: color})
def MaakKleurRec(pcontext):
  subj = pcontext.match[1].concept
  col = pcontext.match[2].concept
  yield pcontext.transientconcept('changecolor({}, {})'.format(subj, col),
        {instanceof: changecolor, subject: subj, goal: col})

v = {'word':'zet'}
subj = {instanceof_t: obj, istype:None}
pp = {instanceof: location, 'prepphrase':True, 'preptype':location}
@parser(v, subj, pp)
@parser(v, subj, pp, {'word':'neer'})
@parser(v, pp, subj)
@parser(v, pp, subj, {'word':'neer'})
def ZetRec(pcontext):
  if pcontext.match[1].concept.testrel(instanceof, location):
    loc = pcontext.match[1].concept
    subj = pcontext.match[2].concept
  else:
    assert pcontext.match[2].concept.testrel(instanceof, location)
    subj = pcontext.match[1].concept
    loc = pcontext.match[2].concept
  yield pcontext.transientconcept('changelocation({}, {})'.format(subj, loc),
        {instanceof: changelocation, subject: subj, goal: loc})

@parser({'word':'maak'}, {instanceof_t: obj, istype:indeterminate, 'constraint_instance':None})
@parser({'word':'maak'}, pp, {instanceof_t: obj, istype:indeterminate, 'constraint_instance':None})
@parser({'word':'maak'}, {instanceof_t: obj, istype:indeterminate, 'constraint_instance':None}, pp)
def MaakObjRec(pcontext):
  loc = None
  subj = pcontext.match[1].concept
  if len(pcontext.match) == 3:
    if pcontext.match[1].concept.instanceof == location:
      loc = pcontext.match[1].concept
      subj = pcontext.match[2].concept
    else:
      subj = pcontext.match[1].concept
      loc = pcontext.match[2].concept
  if loc:
```

```python
    subj = pcontext.copytransient(subj)
    try_delrel(isat, subj)
    rel(isat, subj, loc)

  yield pcontext.transientconcept('createobject({})'.format(subj),
        {instanceof: createobject, subject: subj})



@parser({instanceof_t: obj, istype:None}, {'word':'is'}, {instanceof_t:{color, location}, istype:None})
def PredicateRec(pcontext):
  subj = pcontext.match[0].concept
  direct_obj = pcontext.match[2].concept
  rels = []
  for reltype, targets in subj.sjidx.items():
    if direct_obj in targets:
      rels.append(reltype)
  if not rels:
    if direct_obj.testrel(instanceof_t, color):
      rels.append(iscolor)
    elif direct_obj.testrel(instanceof_t, location):
      rels.append(isat)
  for reltype in rels:
    yield pcontext.transientconcept('pred {}({}, {})'.format(reltype, subj, direct_obj),
        {instanceof: predicate, subject: subj, relation: reltype, target: direct_obj})


def constraintmatch(con, constraint):
  assert constraint.testrel(istype, indeterminate)
  for type, target in constraint.gettargets():
    if not type == istype and not type in transitive_relations and not con.testrel(type, target):
      return False
  return True

def findmatches(constraint):
  assert constraint.testrel(istype, indeterminate)
  type, targ = next((type, targ) for type, targ in constraint.gettargets()
                    if type.testrel(instanceof_t, physical_prop))
  for c in targ.getreverserels(type):
    if not c.testrel(istype, indeterminate) and constraintmatch(c, constraint):
      yield c


@parser({instanceof:kategory})
def KategoryConstraintRec(pcontext):
  kat = pcontext.match[0].concept
  yield pcontext.transientconcept("constraint({})".format(kat), {instanceof:kat, istype:indeterminate}), dict(pcontext.match
    [0].features, constraint_instance=True)

@parser({'determiner':True, 'definite':True, 'indexical':None}, {'
    constraint_instance':True})
def InstanceRec(pcontext):
  g1, g2 = (m.features['gender'] for m in pcontext.match)
  if g1 != g2 and g1 != None:
    return
  for c in pcontext.match[1].concept.instanceof.fullextension():
    if not c.testrel(istype, indeterminate) and constraintmatch(
                c, constraint=pcontext.match[1].concept):
      yield c, {}, activation[c]

@parser({'determiner':True, 'definite':True, 'indexical':None, '
    gender':'mv'}, {'combined_adjectival':True})
@parser({'determiner':True, 'definite':True, 'indexical':None, '
    gender':'mv'}, {'adjectival':True})
def ImplicitInstanceRec(pcontext):
  for c in findmatches(pcontext.match[1].concept):
    yield c, {}, activation[c]

@parser({'determiner':True, 'definite':True, 'indexical':True, '
    gender':'mv'}, {'combined_adjectival':True})
@parser({'determiner':True, 'definite':True, 'indexical':True, '
    gender':'mv'}, {'adjectival':True})
def ImplicitIndexicalInstanceRec(pcontext):
  indexicals = pcontext.match[0].referents()
  for c in findmatches(pcontext.match[1].concept):
    if c in indexicals:
      yield c, {}, prob_or(indexicals[c], activation[c])


@parser({'determiner':True, 'definite':True, 'indexical':True}, {'
    constraint_instance':True})
def IndexicalInstanceRec(pcontext):
  g1, g2 = (m.features['gender'] for m in pcontext.match)
  if g1 != g2 and g1 != None:
    return
  indexicals = pcontext.match[0].referents()
  for c in pcontext.match[1].concept.instanceof.fullextension():
    if c in indexicals and constraintmatch(c, pcontext.match[1].
        concept):
      yield c, {}, prob_or(indexicals[c], activation[c])


@parser({'determiner':True, 'definite':False}, {'
    constraint_instance':True})
def IndetInstanceRec(pcontext):
  con = pcontext.copytransient(pcontext.match[1].concept)
  con.name = 'newobject'
  yield con


```

```
@parser({instanceof_t:obj, istype:None}, {instanceof: location, '
    prepphrase':True, 'preptype':location, istype:None})
def InstancePPCombiner(pcontext):
  if pcontext.match[0].concept.testrel(isat, pcontext.match[1].
      concept):
    yield pcontext.match[0].concept, pcontext.match[0].features,
        pcontext.match[0].p * pcontext.match[1].p

@parser({instanceof_t:obj, istype:indeterminate, '
    constraint_instance':None}, {instanceof: location, 'prepphrase
    ':True, 'preptype':location, istype:None})
def IndetInstancePPCombiner(pcontext):
  con = pcontext.copytransient(pcontext.match[0].concept)
  try_delrel(isat, con)
  rel(isat, con, pcontext.match[1].concept)
  yield con, pcontext.match[0].features, pcontext.match[0].p *
      pcontext.match[1].p

@parser({'adjectival':True, 'combined_adjectival':None}, {'
    constraint_instance':True})
def ConstraintCombiner(pcontext):
  con = pcontext.transientconcept(
          'combined_constraint({})'.format(pcontext.match[1].
              concept.instanceof),
          {istype:indeterminate})
  for type, target in chain(pcontext.match[0].concept.gettargets()
      ,
                              pcontext.match[1].concept.gettargets()
                                  ):
    if type in singleton_relations and (pcontext.match[0].concept.
        getrels(type) and
                                            pcontext.match[1].concept.
                                                getrels(type)):
      # Whoops, there can only be one instance of this type of
          relation, we reject e.g.
      # "Het groene groene vierkant".
      return
    if not type == istype and not type in automatic_relations:
      rel(type, con, target)
  yield con, dict(pcontext.match[1].features, constraint_instance=
      True)

@parser({'adjectival':True}, {'adjectival':True})
def AdjectivalCombiner(pcontext):
  m0, m1 = pcontext.match
  if m0.features['gender'] and m0.features['gender'] != m1.
      features['gender']:
    return
  con = pcontext.copytransient(m0.concept)
  copy_transient_concept_features(m1.concept, con)
  f = dict(m0.features)
  f.update(m1.features, combined_adjectival=True)
  yield con, f
```