Fachrichtung 5.7 Computerlinguistik
Universität des Saarlandes

On Modeling Meaning Shifts by Relaxing Underspecified Semantic Representations

Diplomarbeit

Angefertigt unter der Leitung von
Prof. Dr. Manfred Pinkal

Kristina Striegnitz

Dezember 1999

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst und keine als die angegebenen Quellen und Hilfsmittel verwendet habe.

Saarbrücken, den 31.12.1999

Kristina Striegnitz

Abstract

The context in which a word appears in natural language often influences its interpretation in such a way that the base meaning of the word is changed or made more specific. Polysemy and metonymy are examples for this phenomenon. These meaning shifts of words can be modeled by augmenting the semantic representation of a natural language utterance with the information that is missing to make the shift in meaning explicit. This information can be provided by linguistic or non-linguistic sources or an interaction of both.

Recently, Egg (1999) has suggested an account of meaning shifts which exploits underspecification methods to yield a monotonic augmentation process. The main idea is to have semantic construction derive a sufficiently relaxed (i.e. made less specific) semantic representation, so that adding the missing information is simply further specification of this representation.

This thesis will examine a treatment of meaning shifts due to systematic polysemy or metonymy within Egg´s framework. We will present a syntax/semantics interface which derives appropriately relaxed semantic representations. To account for meaning shifts these representations can be augmented monotonically with additional information.

We will point out a potential problem for this approach: making underspecified semantic representations less specific may cause overgeneration. However, as we will show, for our applications relaxation is safe, i.e. there is no danger of overgeneration.

The underspecification formalism that we will use throughout this thesis is in the class of tree description languages subsuming dominance constraints. We will distinguish a novel class of subconstraints with a certain structure which powerfully support the type of inferences on dominance and disjointness which we have to make for proving safety of relaxation.

JACK We'll split up the week. You can have lymphoma, tuberculosis and ...

MARLA You take tuberculosis. My smoking doesn't go over at all.

JACK I think testicular cancer should be no contest.

MARLA Well, technically, I have more of a right to be there than you. You still have your balls.

JACK You're kidding.

MARLA I don't know ... am I? I'll take the parasites.

JACK You can't have both parasites. You can take blood parasites ...

MARLA I want brain parasites.

JACK Okay. I'll take blood parasites and organic brain dementia ...

MARLA I want that.

JACK You can't have the whole brain!

MARLA So far, you have four and I only have two!

JACK Then, take blood parasites. It's yours. Now we each have three.

...

MARLA So, we each have three – that's six. What about the seventh day? I want ascending bowel cancer.

JACK I want ascending bowel cancer.

MARLA That's your favorite, too? Tried to slip it by me, eh?

JACK We'll split it. You get it the first and third Sunday of the month.

MARLA Deal.

from: Fight Club, 1999. Directed by David Fincher.

Marla and Jack are discussing who may go to which self-help group.

Contents

1 Introduction **10**

1.1 Meaning Shifts 10

1.2 Semantic Underspecification 12

1.3 Underspecified Representations of Meaning Shifts 14

1.4 Contribution 15

1.5 Structure of the Thesis 16

2 Meaning Shifts **18**

2.1 Meaning Shifts in Lexical Semantics 18

2.2 Modeling Meaning Shifts by Augmentation of the Semantic Representation 22

2.3 Summary 28

3 Approaches to Modeling Meaning Shift **29**

3.1 Dölling 30

3.2 Hobbs 32

3.3 Pustejovsky 34

3.4 Briscoe, Copestake, and Lascarides 36

3.5 Summary 39

4 Semantic Underspecification With Tree Descriptions **41**

4.1 λ-terms as Trees: the Object Language 42
4.2 Graphs as Tree Descriptions: the Meta-Language 44
4.3 Application to Scope Ambiguity 49
4.4 Semantic Construction 50
4.5 Summary 53

5 Modeling Meaning Shifts in an Underspecification Framework 54
5.1 Relaxing Underspecified Semantic Representations 55
5.2 Safety of Relaxation 60
5.3 Proving Satisfiedness of the Safety Criterion 65
5.4 A Safe Syntax/Semantics Interface 68
5.5 Summary 74

6 Lots of Questions and Some Answers 75
6.1 When to Add a Shift Operator 76
6.2 Where to Add the Shift Operator 79
6.3 What Shift Operator to Add 81
6.4 Summary 86

7 Conclusion 88

A Proofs 90
A.1 Properties of Fragments 90
A.2 Properties of Relaxation and Projection 92
A.3 The Safety Criterion 96
A.4 Properties of Chains and Cascades of Fragments 98
A.5 A Safe Syntax/Semantics Interface 102

Acknowledgments

First of all, I would like to thank my professor Manfred Pinkal for all the support and encouragement I received from him during my studies in Saarbrücken. He has created a stimulating research environment at his department.

I am indebted to Markus Egg who has initiated my research in this area and has helped me in many interesting discussions with his broad knowledge in the field of semantics.

I am extremely grateful to Joachim Niehren who has always shown a contagious enthusiasm. Working with him has been very inspiring and has considerably influenced this thesis.

Special thanks go to Alexander Koller for his incessant interest in my work which helped me keep my confidence in what I was doing, for the many times I received help and advice from him, and for the uncountable inspiring discussions (not only on the topic of this thesis). His numerous helpful and detailed comments on earlier drafts of this thesis have contributed much to its present form.

Chapter 1

Introduction

In this chapter, we give an overview over the topic of this thesis and introduce the main concepts, such as meaning shift, semantic underspecification, tree descriptions, and relaxation of tree descriptions.

1.1 Meaning Shifts

Lexical semantics studies the meaning of words.An interesting problem in lexical semantics is that sometimes the original meaning of a word does not match the context, as in the following examples, which are discussed in great detail in (Nunberg 1995) and (Pustejovsky 1995).

(1.1) (a) I'm parked out back.

(b) Mary began a book.

In 1.1 (a) *parked out back* requires a vehicle as its argument, but what is given is a person, namely the speaker. Furthermore, it is striking that this sentence can be felicitously uttered by somebody standing right next to you inside the house if only his car is parked out back. Since it seems neither

plausible to assume a second base meanings for *I* (denoting the speaker´s car) nor for *parked* (denoting a property of human beings instead of vehicles), we have to assume that the base meaning of either one is *shifted* in the course of semantic processing.

In 1.1 (b) things are similar: One can only begin to *do* something. *begin* should therefore require one of its arguments to be an individual (corresponding to the agent) and the other one to be something expressing the eventuality or action which is begun. Let's assume here for the purpose of this presentation that this is reflected by an argument of type t.[1] The semantic type of *begin* is then $\langle t, \langle e, t\rangle\rangle$. *book* clearly denotes a property of individuals; it is of the semantic type $\langle e, t\rangle$. In combining the semantic representations of the lexical items, a simple semantic construction, which considers only information explicitly mentioned in the utterance, would derive the following semantic representation.

$$a(book)(\lambda x.begin(x)(mary))$$

It contains a type conflict, since *book* constrains the x to be of type e, while *begin* requires it to be of type t.[2] In fact, the usual way to understand this sentence is as *Mary began to read a book* (or perhaps *Mary began to write a book*, if Mary is known to be an author). So a human hearer understands things which are not explicitly uttered, but which can be inferred from the context and general knowledge about the world. The semantic representation is *augmented* by this additional information, as shown below (new parts, w.r.t. the semantic representation based on purely linguistic information, are underlined).

$$a(book)(\lambda x.begin(\underline{read(x)(mary)})(mary))$$

[1] In a Davidsonian representation *begin* would probably require an event type as argument. This would still lead to a type or sort conflict and not significantly affect the point that we want to make here by using this example.

[2] One possible way out would of course be to assume more than one lexical entry for the verb *to begin*, which fills in an expression expressing that *something is done* to the argument individual in case of an NP object. This would lead to a semantic representation as $a(book)(\lambda x.begin(ACT\_ON(x)(mary))(mary))$. However, it has been argued elsewhere (Pustejovsky 1995) extensively, why this is not a desirable solution. We will briefly return to this question in Chapter 2.

This *augmentation of the semantic representation* due to knowledge which is not made explicit in the utterance is what causes the *meaning shift*. We will call the material with which the semantic representation is augmented the *shift operator*.

1.2 Semantic Underspecification

In the past years, underspecification has become a popular device for describing cases of semantic ambiguity in such a way that the exhaustive enumeration of all readings is avoided (see. e.g. van Deemter and Peters (1996)). Here, we will adopt the view on semantic underspecification put forward by Pinkal (1995). He proposes to radically underspecify in all cases in which, for some reason or another, information is not complete. There are many reasons for incomplete information in natural language (ambiguities on all levels of linguistic processing, ungrammatical utterances in spoken language, disturbed acoustic signal in noisy environments etc.). Traditionally, linguistic processing has fully specified representations of the semantic content of an utterance as its goal. In cases where this content cannot be completely determined by linguistic information alone, the result has to account for all possible readings and is therefore a *set* of semantic representations. This set may easily become quite large, which makes further processing inefficient. Underspecification tries to overcome this problem by assuming the result of linguistic processing to be *one* underspecified representation. This underspecified representation is built according to the amount of information available at the time of processing. What is not known for sure, is not considered. In this way, the result describes all readings the utterance in question could possibly have and, of course, it should describe only these.

We will now give an informal introduction to the basic ideas of underspecification. In Chapter 4 we will present an underspecification formalism in full detail.

The main idea in most underspecification formalisms is to take the representation language that we use for fully specified readings as the *object language*. We can then devise a *meta language* which lets us talk about expressions of the object language in an underspecified way. That is, one expression of the meta language describes a set of object language expressions without enumerating them. The underspecified meta language representation can then be

further specified when new information is gained, possibly from non-linguistic sources. Further processing now has to consider only one representation and enumeration of all readings is delayed as long as possible or necessary.

To illustrate how underspecification works, we will use the following well-known example containing a scope ambiguity:

(1.2) Every man loves a woman.

This sentence contains two scope bearing elements, namely the two quantifiers (*every man* and *a woman*). Their ordering with respect to scope is free. This gives rise to two possible readings, which are as follows:

$$every(man)(\lambda x_1.a(woman)(\lambda x_2.love(x_1)(x_2)))$$
$$a(woman)(\lambda x_2.every(man)(\lambda x_1.love(x_1)(x_2)))$$

In the first case, the universal quantifier has wide scope over the existential quantifier and in the second case, it is the other way round. But both readings are built from the same "semantic material", i.e. the following "fragments":

1) $every(man)(\lambda x_1.\bullet)$ **2)** $a(woman)(\lambda x_2.\bullet)$ **3)** $love(x_1)(x_2)$

A $\bullet$ intuitively stands for a hole into which additional material can be filled. The only difference between the two readings is that the fragments are composed in different ways. Figure 1.1 describes the possible ways of arranging the fragments into one formula. The dotted lines express that fragment 3) has to be a part, proper or not, of the material which is filled into the holes of the fragments 1) and 2). This leaves two possibilities of arranging the fragments: firstly, fragment 3) is filled into the hole of fragment 1), which is then filled into the hole of fragment 2); secondly, fragment 3) can be used to fill the hole of fragment 2), which then goes into the hole of fragment 1).

Although the details may differ, in principle most underspecification formalisms work in a very similar fashion (e.g. Bos 1996; Reyle 1993). In particular, the Constraint Language for Lambda Structures (CLLS) (Egg et al. 1998), which is the underspecification formalism that we will use throughout this thesis, does. CLLS is a tree description language, i.e. it is a meta language for talking about sets of trees, which are expressions of the object

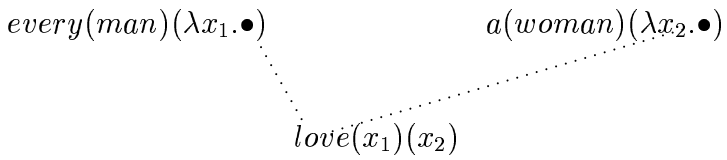


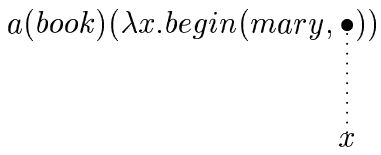
Figure 1.1: Underspecified representation of *Every man loves a woman*

language. Since logical formulas can be represented by trees it can be used as a formalism for semantic underspecification. Trees are described in CLLS by a set of constraints on the relative position and the labeling of their nodes. Underspecification is introduced by dominance constraints which state that two nodes n_1, n_2 are either equal or one is "above" the other in the tree. This leaves unspecified whether there are any and if how many nodes on the path from n_1 to n_2. CLLS is introduced in Chapter 4.

1.3 Underspecified Representations of Meaning Shifts

Markus Egg (Egg 1999) proposes a view on phenomena of meaning shift which exploits underspecification mechanisms as presented in the previous section. His idea is to introduce dotted edges as we saw above at all places where it could be necessary to introduce a shift operator. This yields a *relaxed* semantic representation; the representation is less specific and allows for new material to be included into the representation. Remember that a dotted edge from a hole to a fragment means that the only restriction on what's used to fill the hole is that it has to contain the fragment. So the dotted edges can be taken to be gaps in the semantic representation, which allow to introduce new material (such as a shift operator). If the introduction of a shift operator is not necessary, the gap can be closed, i.e. the hole is filled with the fragment it is connected to by the dotted edge.

In Egg's approach, the (relaxed) semantic representation which semantic construction derives for *Mary began a book* is:

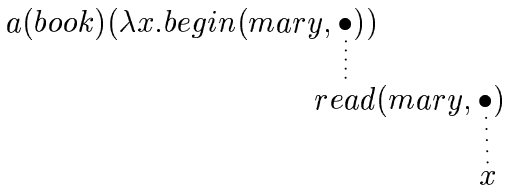


If we just closed the gap, we would end up with a type conflict, as explained in Section 1.1.

However, we now have the option of filling an appropriate shift operator, e.g.

$$read(mary, \bullet),$$

into the gap. This would produce the following semantic representation:



All the gaps in this representation can be closed without running into a type conflict, and the result is the intended reading, namely

$$a(book)(\lambda x.begin(mary, read(mary, x))).$$

This approach to modeling meaning shifts compiles the destructive part of it, namely breaking up the semantic representation to be able to fill in the missing information, into the syntax/semantics interface. The shift operator can then be added by simply instantiating gaps.

This approach will be presented in more detail in Chapter 5.

1.4 Contribution

Relaxing underspecified semantic representations means making an underspecified representations even less specific. We will point out that this may lead to structurally new solutions, i.e. solutions which differ from solutions for the unrelaxed representation not only at the relaxation site. In cases

where relaxation was done solely for the purpose of augmenting the semantic representation with *new* material, a problem of overgeneration arises. The central idea of Egg's (1999) approach to modeling meaning shifts relies on relaxing underspecified semantic representations for augmentation by the shift operator. This means that Egg's approach is in principle subject to the same danger of overgeneration.

We will investigate this problem in the framework of CLLS, where underspecified representations are expressed by tree descriptions subsuming dominance constraints. We will formulate a criterion which ensures that an underspecified representation can be relaxed without adding unwanted readings. In developing tools for proving satisfiedness of the criterion, we will distinguish a substructure of constraints with particularly pleasant properties, that is of importance not only in connection with our particular problem.

We will apply the criterion to analyses of polysemy and metonymy which are along the lines of Egg's approach. We will present a syntax/semantics interface that derives relaxed underspecified semantic representations and thus provides the basic structures for a treatment of meaning shift based on Egg's proposal. We will then prove that this syntax/semantics interface never leads to any overgeneration of the type discussed above.

Relaxing the semantic representations is only the very first step in the process of meaning shift. It will be in the center of this work, though. We will only very briefly discuss further questions, such as how to find the appropriate shift operator, which is a problem that is still largely unsolved. We will show how previously suggested solutions, such as qualia structure (Pustejovsky 1995), dotted or hybrid sorts (Pustejovsky 1995; Dölling 1994), intersortal relations (Dölling 1994), are not in opposition to Egg's approach, but could be incorporated.

1.5 Structure of the Thesis

In Chapter 2 will introduce the linguistic background of this thesis. We will examine the phenomena of lexical semantics we are interested in and distinguish *polysemy*, *metonymy*, and *metaphor* as phenomena of meaning shift. We will then briefly address the main questions which have to be dealt with by a process of meaning shift.

In Chapter 3, we will present previous work that has been done on meaning shifts. We will discuss four approaches. Dölling (1994), more explicitly than the others, explains meaning shifts by augmentation of the semantic representation with a shift operator. Hobbs et al. (1993) suggest to model meaning shift by inference on world knowledge, while Pustejovsky (1995) relies only on lexicalized information. Copestake and Lascarides (1998), finally, argue that meaning shifts are influenced by both, lexical as well as pragmatic, knowledge. They use a default logic and inference on world knowledge and discourse structure to capture this.

Chapter 4 will present the formal background of this thesis. It will introduce CLLS (Constraint Language for Lambda Structures), the underspecification formalism which we will use throughout the rest of this thesis. We will illustrate its use for linguistic applications by describing the application to scope ambiguity. Furthermore, we will present a syntax/semantics interface which compositionally derives underspecified semantic representations.

In Chapter 5 Egg's approach to meaning shift is developed in this formalism. We will show that relaxation of underspecified semantic descriptions, the main strategy in Egg's approach, can in principle lead to overgeneration, but in the case of its application to modeling meaning shifts never does. In developing tools for proving this we will discover some interesting novel properties of a certain subclass of constraints.

In Chapter 6 we will show how previously suggested solutions, such as qualia structure (Pustejovsky 1995), dotted or hybrid sorts (Pustejovsky 1995; Dölling 1994), intersortal relations (Dölling 1994), could be incorporated into Egg's approach.

Finally, Chapter 7 is devoted to open questions and future work.

Chapter 2

Meaning Shifts

In this chapter I will examine more closely those phenomena of lexical semantics which we are concerned with in this thesis. After having classified them, we will look at different aspects of a treatment based on the insertion of a shift operator.

2.1 Meaning Shifts in Lexical Semantics

Cases where a word can be used to mean different things depending on the context it appears in are abundant in lexical semantics. Four classes of phenomena have received particular attention: *homonymy*, *polysemy*, *metonymy*, and *metaphor*. We will now introduce the properties that are usually ascribed to these phenomena and then discuss which of them are cases of *meaning shift*.

2.1.1 Homonymy and Polysemy

Homonymy and *Polysemy* are cases of *lexical ambiguity*, where one word form has several different meanings. The examples in 2.1 illustrate the lexical

ambiguity of the words *bank* and *newspaper*.

(2.1) (a1) Susan walks along the *bank* of the river.

(a2) The *bank* was robbed last night.

(b1) The *newspaper* was founded in 1891.

(b2) John spilled coffee on the *newspaper*.

(b3) Mary is reading a *newspaper*.

In 2.1 (a1) *bank* refers to the shore of a river, while in 2.1 (a2) it refers to a financial institution. In all of these cases the context disambiguates which is the intended meaning. Similarly, contextual restrictions imply that 2.1 (b1) is talking about an publishing company, while in 2.1 (b2) John spilled coffee on a paper object, and in 2.1 (b3) the word *newspaper* is carrying a notion of information.

The definitions of homonymy and polysemy which can be found in the literature vary and imply different classifications of some examples of lexical ambiguity. Fries (1980) distinguishes approaches with a diachronic perspective from those with a synchronic perspective. Under a diachronic perspective the distinction between homonymy and polysemy is made according to etymological criteria: if the different senses a word has are etymologically related one speaks of polysemy, while the different senses of homonymic words are derived from etymologically different stems and word forms only accidentally coincide. However, people are often not aware of etymological relationships, so that such a criterion is not adequate under a synchronic perspective. Under a synchronic perspective we can distinguish between objective and subjective criteria to characterize homonymy and polysemy. As "hard" criteria morpho-syntactic and morpho-phonological differences (e.g. the difference of gender between the German words for *pine tree*, i.e. *die Kiefer*, and *jaw*, i.e. *der Kiefer*) can be used. If such a difference can be found, the ambiguity is classified as homonymy. In addition to these objective criteria, the degree of difference between different senses of an ambiguous word and the degree of unity they exhibit is also often used. These are subjective criteria, so that the distinction between homonymy and polysemy becomes fuzzy.

As for the examples in 2.1, *bank* is a clear case of homonymy and the ambiguity illustrated by 2.1 (b2) and 2.1 (b3) is a clear case of polysemy. The

ambiguity between the *company*-reading of *newspaper* on the one hand and the other two readings on the other hand, however, could, from a synchronic point of view, be argued to be an example of homonymy. From my subjective point of view the *company*-reading of newspaper and the *information bearing paper object*-reading are sufficiently different and do not exhibit much unity (although they are of course related). In fact there have been differing opinions on how to treat *newspaper* (cf. (Copestake and Briscoe 1995)).

An interesting fact about polysemy is that it might give rise to a polymorphous and vague behavior as illustrated by the following examples.

(2.2) (a) Mary painted the *door* blue.

(b) Mary walked through the *door*.

(c) Mary opened the *door*.

(d) A dwarf stepped out of the blue *door*.

In 2.2 (a) *door* is used to make reference to a physical object, while 2.2 (b) talks about an aperture. Both of these aspects seem to be involved in 2.2 (c). It is not possible to unambiguously decide whether we are talking about the physical object or the aperture sense of *door*. A similar effect is produced by 2.2 (d). Here, the meaning of the word *door* which is only realized once on the surface seems to vary from one semantic occurrence to another: *blue* requires the physical object aspect and *step out of* the aperture one.

2.1.2 Metonymy and Metaphor

Metonymy and *Metaphor* are rhetorical stylistic devices which allow to refer to an object or event by using words or expressions that are in some kind of relationship to the described object or event, but do not exactly match the properties of the object under a conventional interpretation. Consider the following examples:

(2.3) (a) Goethe is widely read. *(for: Many people read books by Goethe)*

(b) John is wearing silk. *(for: John is wearing clothes made out of silk)*

(c) This pig never washes. *(for: This person never washes and therefore has some properties usually ascribed to pigs.)*

Bartsch (1984) describes metonymy and metaphor as phenomena which trigger a non-literal interpretation of certain words or expressions. Some properties which would be implied by a literal interpretation might be totally eliminated, while others are strengthened. In her analysis metaphor relies on a similarity relation between the literal meaning of an expression and what it is used to refer to, i.e. they share some (but not necessarily all) characteristics. If, on the other hand, a factual relationship, such as a spatial, temporal, or causal relation ship, is exploited, she talks about metonymy. Under this definition Examples 2.3 (a) (product/producer relationship) and 2.3 (b) (material/product relationship) are cases of metonymy, while 2.3 (c) is a case of metaphor.

It should be noted that the use of such rhetorical devices is not simply a substitution of one expression by another while the meaning does not change. Usually, the metaphorical or metonymic expression carries further connotations and aspects which all influence the picture which is evoked. Consider e.g. the following example due to Cruse (1991).

(2.4) The huge locomotive snorted and belched its way across the plain.

In addition to making a statement about the noises the locomotive produces, the sentences associates it with a "gigantic" animal, which connotes further characteristics.

2.1.3 Meaning Shift

We defined meaning shift in the introduction to be a process that changes the base meaning of a word and we claimed that it could be modeled by augmenting the semantic representation of a given utterance with additional material (a shift operator).

Metonymy and metaphor most clearly seem to be cases of meaning shift, since one of their major features is that the literary meaning (the base meaning) of an expression is modified. Actually, the examples we discussed in the

introduction as a motivation for an operation of meaning shift (*I am parked out back* and *Mary began a book*) are cases of metonymy.

So by meaning shift we can capture the relationship that holds between the literal meaning of a metonymically or metaphorically used expression and the object or event it is used to describe. We argued above, however, that the usage of such rhetorical devices may have further effects. So, for example the pure existence of some kind of relationship doesn't justify the metonymic or metaphoric usage. We will briefly consider such restrictions in the next section. As for others of these "additional" effects, we will not go into them here, since they were not in the center of the work presented in this thesis.

The essence of the various definitions which have been suggested for homonymy and polysemy is that the different senses of a homonymic expression are, in contrast to the case of polysemy, rather unrelated and should be considered as different lexemes. Cruse's (1991) definition makes this precise. He regards one lexeme which has several senses as polysemic and different lexemes which have identical word forms as homonymic. This means that finding the right interpretation of a homonymic word is a question of retrieving the right lexical entry (no meaning shift is involved), while in the case of polysemy there has to be a mechanism which selects a meaning according to contextual and other restrictions. So this is not really a case of shifting the meaning of a word, but rather of making it more precise. We will see later (in Chapter 6), however, that this is not necessarily a totally different operation and can be modeled by augmenting the semantic representation.

2.2 Modeling Meaning Shifts by Augmentation of the Semantic Representation

We defined *augmentation of semantic representations* as adding material to a semantic representation. We will use this operation to model meaning shifts and the material that we will augment the semantic representation with is an appropriate shift operator. This immediately brings up three questions. First, under which circumstances is adding a shift operator necessary, i.e. what triggers meaning shifts? Second, where in the semantic representation is the shift operator to be filled in? And finally, what *is* an appropriate shift operator and where does the missing information come from? In the

remainder of this chapter, we will address each of these three questions in turn and examine them with respect to examples of systematic polysemy and metonymy.

2.2.1 When to Add a Shift Operator

In all examples we have seen so far, meaning shift was necessary to circumvent sort or type conflicts in the semantic representation. Remember, for example, once again the sentence *I am parked out back*, where *parked out back* requires an argument of sort *vehicle* while what is given is of sort *human*.

In general, meaning shift is triggered by such a lurking sort or type conflict. However, sometimes the original meaning of some word has to be adapted to fit the context, even without the occurrence of such a conflict. The following utterance, made by my aunt to her daughter, who had just taken a picture of the family, is an example.

(2.5) Ich hoffe, du hast uns nicht die Köpfe abgeschnitten.
'I hope you haven't cut off our heads.'

There is no sort or type conflict involved, but world knowledge gives clear preference to the reinterpreted reading in which *our heads* does not refer to the heads of the real persons, but to their picture.

Besides, there are also cases in which meaning shift is not possible, as in the following example given by Nunberg (1995).

(2.6) I was once driven by Jean Gabin.

2.6 sounds odd if the intended meaning is that it is the speaker's car which *was once driven by Jean Gabin*. In principle, there is a shift operator making it possible to use a person to refer to his car (cf. *I am/The old man is parked out back*), but here it is not available. It is still an open problem to explain what restricts meaning shift.

Nunberg tries to capture these restrictions by requiring that what is said about the transferred meaning of a word has to be *noteworthy* of the original meaning, it has to somehow classify the object referred to by the original

meaning in the current situation or in general. Nunberg argues that having a vehicle which is parked out back may classify a person in certain situations while having a car that was once driven by Jean Gabin usually doesn't. So *I am parked out back* is acceptable while *I was once driven by Jean Gabin* is not.

(Copestake and Briscoe 1995) found that another factor restricting meaning shift is blocking of synonyms. They argue that 2.7 (a) and 2.7 (b) are highly marked in contrast to 2.7 (c), because of the existence of the words *pork* and *beef*.

(2.7) (a) ?? Sam ate pig.
(b) ?? Sam ate cow.
(c) Sam ate lamb.

There is no clear dividing line between acceptable and unacceptable cases of meaning shift. On some examples people's opinions differ. Also, the range of acceptability can be expanded considerably by placing questionable examples in an appropriate context.

2.2.2 Where to Add the Shift Operator

We already mentioned once or twice that there are sometimes several possibilities of applying the shift operator to predicates or parts of the semantic representation. Is, for instance, in the case of *I am parked out back* the meaning of *I* shifted to mean *the speaker's car* or is the meaning of *parked out back* shifted to denote a property of a person instead of a property of a vehicle?

Nunberg (1995) argues that in this case, it must be the predicate introduced by the verb to which the shift operator is applied, because 2.8 (a) is possible, while 2.8 (b) is not.

(2.8) (a) I am parked out back and have been waiting for 15 minutes.
(b) * I am parked out back and may not start.

If the meaning of *I* was shifted it should be possible to say more about the speaker's car in coordinate verb phrases, which is not possible as shown in

2.8 (b). On the other hand, it shouldn't be possible in this case to conjoin predicates describing the speaker as in 2.8 (a).

However, there are also examples for the other variant, where the meaning of a common noun is shifted. In these cases, coordination with statements about the original meaning of the noun is not possible, while information about the transferred meaning can be conjoined. Nunberg gives by the following example:

(2.9) (a) The ham sandwich at table 6 is getting impatient.

(b) * The ham sandwich at table 6 is getting impatient and stale.

(c) The ham sandwich at table 6 has been calling the waiter for 5 minutes now and is getting really angry.

Examples involving quantification can also serve as tests. Consider the following sentence.

(2.10) Most British authors are widely read.

Now assume that there was only one British author who is famous all over the world. All of his books are widely read. He was very productive and wrote more books than all the other British authors taken together, i.e. more than 50% of all books by British authors are written by this one very famous guy. In this situation 2.10 is not true. Now assume that instead of this one very productive author, most of the other authors were very famous, though only some of their books are widely known. In this case 2.10 can be uttered felicitously. Summarizing, 2.10 becomes true, if most British authors are very famous and each of them has written some books which are now widely read. 2.10 is not true, if most of the books by British authors are widely read and all of these widely read books were written by only one author. Hence, the quantification of *most* in sentence 2.10 is over British authors, i.e. the original meaning, and not over books written by British authors. This indicates that the shift operator was applied to the predicate introduced by the verb, and not to the predicate introduced by the noun. The argumentation goes as follows: If the meaning of the noun instead of the verb was shifted, *most* would be applied to the shifted meaning, and therefore quantification would have to be over the books instead of the authors.

Now consider the following sentence in the situations given below:

(2.11) Most ham sandwiches complained about ...

Imagine that there are 10 guests in the restaurant who ordered one ham sandwich each. Furthermore, there is this big, fat guy who ordered 20 ham sandwiches. Sentence 2.11 is not true, if only the big, fat guy complained, but it does turn out true, if all of the other ham sandwich orderers complained. So in this example, *most* does quantify over the shifted meaning, namely orderers of ham sandwiches, instead of over the original meaning, the ham sandwiches. So things are just the other way round, which implies that here the meaning of the noun and not of the verb was shifted.

Shifting the meaning of the verb seems to be the standard case, while shifting common nouns appears to be restricted to use in special settings where they are quite conventionalized, such as using dishes to refer to their orderers in restaurant settings, instruments to refer to their players in orchestra environments, or diseases to refer to those who have them in hospitals.

2.2.3 What Shift Operator to Add

The opinions on how the appropriate shift operator is found differ considerably. The main question is how much of this process is conventionalized and can therefore be lexicalized and how much is driven by context or pragmatic knowledge.

In examples of systematic polysemy like 2.2 the number of possibilities for shifting the meaning of *door* is limited. The different meanings seem to be actually *contained* in the word, and the sort conflict can be resolved by just *picking* one of them. So, all a shift operator has to do in these cases is to project the appropriate one of the lexically determined senses.

In the examples of metonymy in Section 2.1, some knowledge about the world (for instance the knowledge that people often own cars) is needed to infer the right shift operator. But there are also examples of metonymic meaning shifts which exhibit a more conventionalized behavior. Consider e.g. the following sentences:

(2.12) (a) Mary began a book.

(b) Mary finished a book.

(c) Mary enjoyed a book.

Here, the missing information, namely what is done to the book, seems to be determined mainly by the knowledge one has about books. *Mary began the book* is understood as *Mary began doing to the book what is normally done to books, namely reading*. Copestake and Lascarides (1998) note that, in contrast, the examples in 2.13 are all odd.

(2.13) (a) * John began a chair. *(sitting on)*

(b) * John began a tunnel. *(driving through)*

(c) * John began a film. *(watching)*

This is interesting, because *sitting on*, *driving through*, or *watching* should be connected with *chair*, *tunnel*, or *film* respectively just as closely as *reading* is connected with *book* and it is not clear how the involved events or their relationships to their respective objects differ[1]. This suggests that the question which eventualities are available for metonymic meaning shifts cannot be answered by pragmatics alone. Some kind of lexical licenses are needed, which specify e.g. that books can be read, but not that tunnels can be driven through.

In some cases these lexically determined or conventionalized meaning shifts can be overridden by contextual information and world knowledge, though. Consider the discourse given in 2.14 (a). Verspoor (1997) notes that it can be continued by 2.14 (b) or 2.14 (c) but not by 2.14 (d).

(2.14) (a) My goat went nuts last night. He ate everything in his cage.

(b) He particularly enjoyed your book.

(c) He began on your book at 9pm.

(d) * He began your book at 9pm.

[1]See for example (Verspoor 1997), where restrictions of this type are examined.

Verspoor (1997) argues that *begin the book* can only be shifted to *begin reading the book* or to *begin writing the book*, while *begin on the book* is more flexible and its interpretation actually depends on the context. Verspoor suggests that this is due to the fact that verbs impose restrictions on their arguments specifying which meaning shifts are possible. In the case of verbs like *begin* the shift operator has to be available lexically via the nominal argument, while in the case of *begin on* or *enjoy* anything that can be established by the situation (and has the right sort) is permitted. Verspoor carried out corpus studies of metonymic usages of different verbs which support her hypothesis.

2.3 Summary

We have seen different phenomena of lexical semantics which involve words that have different meanings according to the context they appear in. Of these phenomena we have distinguished *systematic polysemy*, *metonymy*, and *metaphor* as cases of *meaning shift* (but we will not consider *metaphor* any further in this thesis). We found that meaning shift is usually triggered by a type or sort conflict which can be resolved by introducing an appropriate operator, but pragmatic factors may also play a role. There are examples where the shift operator has to be applied to the predicate introduced by the verb and there are other examples where it has to be applied to the predicate introduced by the noun. At first sight, the first case seems to be more common. To find the right shift operator can be quite difficult. For metonymy this usually depends heavily in world knowledge. However, in some cases of metonymy lexical information also plays a role and shift operators in polysemic examples are determined only by lexical information.

Chapter 3

Approaches to Modeling Meaning Shift

In this chapter we present four earlier approaches dealing with meaning shifts. We will start by introducing Dölling's (1994) approach, which can be viewed as a base for Egg's (1999) proposal, as it, more explicitly than the others, accounts for meaning shifts by inserting additional material into the semantic representation. The other three were chosen because of the general difference they show in explaining where the information the semantic representation is augmented with comes from. They differ mainly in to what extent meaning shift is thought to be driven by pragmatics and world knowledge in contrast to lexical knowledge. On the end of the scale are approaches like the one by Hobbs et al. (1993), which rely entirely on pragmatics and world knowledge. The other extreme is represented by Pustejovsky's (1995) approach, where all variations of word meaning are predicted lexically. A middle course is taken by Copestake, Briscoe, and Lascarides (Copestake and Lascarides 1998; Copestake and Briscoe 1995). They allow for some interaction of lexical semantic knowledge and world knowledge.

The main interest of all of these approaches is on how an appropriate shift operator can be found. This is somewhat complementary to Egg (1999),

who does not make any claims with respect to this problem, but is rather interested in how augmentation of semantic representations with additional material can be made possible in the first place. He stresses the point that meaning shift can be viewed as a case of incomplete information which can be treated very naturally by underspecification. He describes such a treatment in an underspecification formalism based on tree descriptions. He assumes some pragmatic component which delivers the disambiguating information, but does not make explicit how such a component could look like.

I will therefore present the approaches, point out problems they might have with respect to the question of *What shift operator to add*, and then compare them with Egg's approach with respect to the mechanisms that allow for adding operators to the semantic representation.

3.1 Dölling

As said above, Dölling (1994) accounts for meaning shifts due to type or sort conflicts by introducing operators into the semantic representation. These operators are applied during the semantic construction process. He distinguishes polysemy and metonymy[1] and treats them slightly different. The shift or rather specialization of meaning in cases of polysemy is effectuated by lexical rules, which may also have syntactic and morphological effects. In this way, he can explain count noun/mass noun alternations, as in Examples 3.1, as a case of polysemy.

(3.1) (a) The bread was round.
(b) John ate some bread.

Dölling assigns polysemic words a so-called *hybrid sort*. Hybrid sorts are linguistic sorts that have no counterpart in the ontology. They correspond to unions of ontological classes that are thought to be disjoint, e.g. apertures and physical objects. Dölling's motivation for hybrid sorts is to capture what we called the polymorphous behavior of polysemic words in Chapter 2. By using the operator $SPEC$ the meaning of the word can be specified to any component of the hybrid sort.

[1]What he calls polysemy and metonymy corresponds to our concepts of systematic polysemy and metonymy, respectively.

The operation of meaning shift in cases of metonymy is not restricted to the application to lexical categories. Furthermore, they never associated with morpho-syntactic changes. Metonymic meaning shift is driven by knowledge about the relations that hold between different classes of an ontology. This general knowledge is encoded into a number of predefined shift operators. Dölling assumes, for instance, a shift operator which, if applied to an individual of sort institution, yields a generalized quantifier that requires its argument to be a predicate specifying a property of a group of people. He also assumes an operator which transforms a predicate specifying a property of a groups of people into one specifying a property of one person. Let's call these operators $OP1$ and $OP2$, respectively. Example 3.2 sketches, how they are applied. I used *the_boston_office* and *call* to stand for the meaning of the NP *the Boston office* and *call* and annotated all parts of the formula with their semantic type and sorts.

(3.2) (a) The Boston office called.

(b) $OP2_{<<e:group\_of\_people,t>,<e:person,t>>}$
$(OP1_{<<e:group\_of\_people,t>,t>}(the\_boston\_office_{e:institution}))$
$(call_{<e:person,t>})$

3.1.1 Comments

Dölling only considers shifting the meaning of nouns. We saw in Chapter 2 that this is not adequate. Dölling's approach has problems with accounting for Example 3.3, for instance.

(3.3) I am parked out back and have been waiting for 15 minutes.

If the meaning of *I* is shifted to refer to the speaker's car, as it is necessary for the combination with *parked out back*, *have been waiting for 15 minutes*, which is a property of the speaker himself, does not fit anymore.

Dölling assumes further that the shift operators are introduced during semantic construction. This requires that the information which determines the type of shift operator to be introduced is available at that time. This is not necessarily the case, when inferences are necessary to determine the shift operator as in the following example.

(3.4) My goat went nuts last night. He ate everything in his cage. He particularly enjoyed your book.

In newer papers Dölling introduces only underspecified operators during semantic construction (cf. (Dölling 1997)). This allows to capture the influence pragmatics and inferences on context and world knowledge may have. Note also that underspecified operators are very similar to Egg's additional dominance edges.

3.2 Hobbs

Hobbs et al. (1993) are not primarily concerned with lexical semantics, but rather suggest an integrated approach to various levels of linguistic processing. A major point is handling incomplete information in natural language, though, and a treatment of metonymy falls out as a by-product.

In their approach, interpreting natural language is viewed as the task of finding the best explanation of why an utterance should be true. They propose a form of abductive inference called weighted abduction for solving this task. In weighted abduction costs are imposed on the assumption of literals and the application of rules. The best explanation is then the one with the lowest costs.

I will now sketch an example, which involves a case of metonymy, to illustrate how abduction works. We will not go into the details of cost calculation, but just give a general impression of how the missing information, the shift operator, is found. In understanding the following sentence, a shift in meaning is necessary to avoid a sort conflict due to the incompatible sorts of the subject *the Boston office*, namely *institution*, and the verb's requirements on its argument, namely *human being*.

(3.5) The Boston office called.

The logical form which Hobbs assumes for 3.5 looks like this:

$\exists x \exists y \exists z \exists e(call(e, x) \wedge person(x) \wedge rel(x, y) \wedge office(y) \wedge boston(z) \wedge nn(z, y))$

Note how the argument of *call* has to be a person. The sort conflict is cushioned, though, by linking x only via $rel(x, y)$ to the semantics of the

subject, which is specified to be an office. $rel(x, y)$ anticipates the places where a meaning shift could occur. The still unspecified relation between the two components of the compound noun is accounted for in a much similar way: they are linked by $nn(z, y)$.

Now assume the following database of background knowledge:

$boston(b_1)$	$\forall y \forall z(in(y, z) \Rightarrow nn(z, y))$
$office(o_1)$	$\forall x \forall y(work\_for(x, y) \Rightarrow rel(x, y))$
$in(o_1, b_1)$	$\forall y(office(y) \Rightarrow \exists x(person(x) \wedge work\_for(x, y)))$

Figure 3.1 illustrates an abductive proof of the logical form of 3.5, where only $call(e, x)$ is assumed. Everything else can be explained by facts or rules from the database.

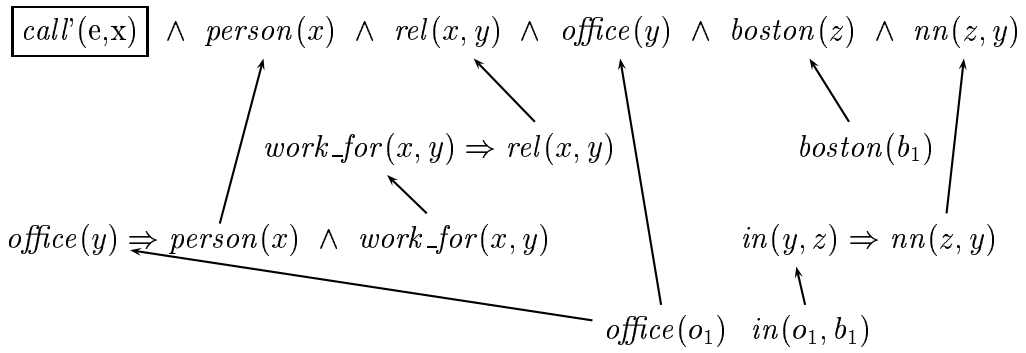


Figure 3.1: Abductively Explaining *The Boston office called*

Costs can interact, so that an appropriate assignment of costs can explain why the prefered readings of 3.6 (a) and 3.6 (b) differ.

(3.6) (a) Max began a book.

(b) Max, the famous author, began a book.

3.2.1 Comments

In (Hobbs et al. 1993) linguistic knowledge stemming from different sources is treated as of the same kind (i.e. it may all be used in the same way to

explain an utterance). However, in Chapter 2 we saw some evidence that different types of information may effect meaning shift in different ways. It is e.g. not obvious how the fact that in Example 3.7 (b) the world knowledge that tunnels are for driving through may not be used to trigger a meaning shift in this direction.

(3.7) (a) Mary began a book. *(reading, writing)*
(b) * Mary began a tunnel. *(driving through)*

As already mentioned above, the *rel*-relation is introduced during semantic construction at those positions where a meaning shift may occur; it is thus very similar to Egg's dotted edges or Dölling's underspecified shift operators.

3.3 Pustejovsky

The backbone of Pustejovsky's (1995) approach to meaning shifts is the idea of allowing lexical items to have a more complex semantic representation than traditionally assumed. The important concepts he introduces are *dot objects* and *qualia roles*.

3.3.1 Dot Objects

Dot objects are Pustejovsky's idea to accomodate for the polymorphic behaviour of systematic polysems. He argues that nouns like *door* don't denote individuals but actually tuples of individuals, so-called dot objects, written as $x \cdot y$. All individuals in the denotation of *door* have the *dotted sort* $physical\_object \cdot aperture$. It is left a bit obscure how exactly dot objects relate to their components and how they fit into the ontology. Special operators, Σ_1 for the first component and Σ_2 for the second, project one of the constituent parts of the dot object.

The following example shows how he suggests to account for the fact that *door* can be an argument of a predicate requiring an object of sort $physical\_object$ as well as of a predicate requiring an object of sort $aperture$[2].

[2]Please note that this formalisation is taken directly from (Pustejovsky 1995). The

(3.8) (a) Mary painted the door blue.
$paint\_blue(\Sigma_1(the\_door))(mary)$

(b) Mary walked through the door.
$walk\_through(\Sigma_2(the\_door))(mary)$

3.3.2 Qualia Roles

Qualia Roles describe selected aspects related to the meaning of a word. In addition to the base meaning these qualia roles are available as an interpretation of the corresponding word. Pustejovsky assumes four qualia roles to be associated with every lexical item.

- The **constitutive** qualia role relates an object to its constituents, i.e. describes its internal structure. For *novel* the constitutive quale states that a novel is a *narrative*, while in the constitutive quale of *dictionary* is something like *list of words*.
- The **formal** qualia role states what distinguishes an object from other objects in its environment. *Novel* and *dictionary* are both distinguished as *information bearing paper object* by the formal qualia role.
- The **telic** qualia role specifies the purpose or function of an object. While novels are usually read, dictionaries are usually consulted.
- The **agentive** qualia roles describes the origin of the object. The agentive quale of *novel* is *write* and of *dictionary compile*.

The qualia structure Pustejovsky gives for the word *book* is presented below. Note that *book* is, like *door*, of a dotted sort. In the qualia structure, reference can be made to all components of the dot object as well as the dot object itself, here: y, of type $physobj$, and x, of type $info$.

$$\begin{bmatrix} \text{FORMAL} & \mathsf{hold(y,x)} \\ \text{TELIC} & \mathsf{read(e,w,x{\cdot}y)} \\ \text{AGENTIVE} & \mathsf{write(e',v,x{\cdot}y)} \end{bmatrix}$$

application of the Σ-operators to *the_door* is probably a bit unconventional, but I think the idea is clear.

The telic qualia role specifies that what is usually done to books is reading them and the agentive qualia structure says that they are created by somebody writing them. Note that both of these actions are done to the dot object of type $physobj \cdot info$. The formal qualia role specifies how the two components of the dot object are related to each other: namely that the physical object holds the information.

Pustejovsky suggests that in cases where the normal interpretation of *book* would lead to a type or sort conflict, any of the qualia roles is also available as a possible interpretation. They can be projected by the application of the operators Q_{Telic}, $Q_{Agentive}$, and so on, as shown in the following example[3]:

(3.9) Mary began a book.
$begin(Q_T(a\_book))(mary)$
$begin(Q_A(a\_book))(mary)$

3.3.3 Comments

Pustejovsky's approach limits meaning shifts to phenomena that can be explained lexically. He notes himself that this is not sufficient to account for the whole range of meaning shifts in natural language semantics.

Similar as Dölling does in (Dölling 1994), Pustejovsky (1995) introduces the fully specified shift operators during semantic construction. That will lead to the same problems as set out above and distinguishes his approach from those that modularize the decisions on where to allow for a meaning shift and what material to add by introducing a certain amount of underspecification.

3.4 Briscoe, Copestake, and Lascarides

Copestake and Briscoe (1995) adopt a somewhat extended and modified version of Pustejovsky's qualia structure. They distinguish two classes of meaning shift phenomena. One of them, *constructional polysemy*, is mainly accounted for by means of this qualia structure. For the other one, *sense*

[3]The same remarks as above hold for this formalization. Furthermore, according to Markus Egg (personal communication), this formalization actually does not work out in the intended way.

extension, they assume lexical rules, which are applicable during parsing and semantic construction.

They also make the interaction of the process which adds the shift operator and semantic construction more precise.

Copestake and Lascarides (1998) then combine this approach with a mechanism to model discourse structure, so that also cases of meaning shift which go beyond conventionality can be captured.

3.4.1 Constructional Polysemy and Sense Extension

In cases of constructional polysemy the meaning associated with a lexical item is specialized by contextual factors. This specialization is essentially the selection of such a qualia role that the requirements of other lexical items are met. Verbs, for example, can coerce their arguments into a different interpretation by selecting one of the qualia roles. Whether verbs are coercing or not is lexically specified. *Begin*, for instance, has four different forms which differ in their subcategorization frame and their ability to coerce their arguments into one of the qualia roles. In constructing a semantic representation for *Mary began a book*, the entry retrieved from the lexicon for *begin* will be specified to one of the coercing versions, which are selecting either the telic or the agentive qualia role of the object as the object semantics.

In cases of sense extension derived senses are created from basic ones by lexical rules. For instance, Copestake and Lascarides (1998) assume a lexical rule for animal grinding, which creates a mass noun usually denoting meat from a count noun denoting an animal. This rule is applied to the lexical entry of *rabbit* to account for the syntactic and semantic behavior of *rabbit* in Example 3.10. The rule is given in Figure 3.2.

(3.10) John eats rabbit regularly.

Copestake and Briscoe (1995) argue that there are several indications that the distinction between sense extension and constructional polysemy is useful. First, the two meanings which are related by sense extension are usually not close enough for there to be a common core meaning which would justify assuming just one lexical semantic structure for both of them. Second, cases of sense extension are often accompanied by syntactic or morphological effects

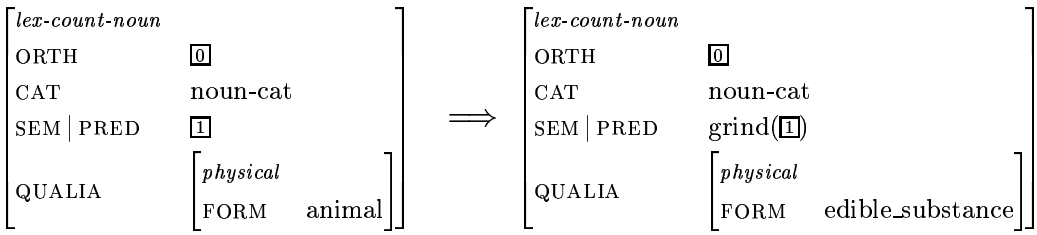


Figure 3.2: Lexical Rule for Animal Grinding

(like e.g. making a mass noun out of a count noun). Following Pustejovsky (1994) Copestake and Briscoe propose to use the possibility of co-predication as an indicator for constructional polysemy. Examples 3.11 (a) and 3.11 (b), where the parts in italics are argument to two different verbs, are examples of co-predication.

(3.11) (a) Mary painted and walked through the door.

(b) The newspaper has been attacked by the opposition and publicly burned by demonstrators.

Apparently, co-predication is more easily possible in cases of constructional polysemy than in cases of sense extension. However, there are cases which Copestake and Briscoe distinguished as constructional polysemy where co-predication is not possible, and there are also cases where co-predication is possible in combination with sense extension.

3.4.2 Interaction With Pragmatics

Now, what about unconventional cases of constructional polysemy, i.e. cases where selection of a qualia role does not provide the appropriate shift operator? An earlier example for this is repeated in 3.12.

(3.12) My goat went nuts last night and ate everything in his cage. He particularly enjoyed your book.

The qualia role of *book* would yield *He particularly enjoyed reading your book* or *He particularly enjoyed writing your book* as possible interpretations of the second sentence of this discourse. This is not what is understood and also a very implausible reading given the fact that goats are usually illiterate.

Copestake and Lascarides (1998) make use of default unification and a theory of discourse structure to account for cases of unconventional constructional polysemy. Very informally, the basic idea is to take the information which is introduced into the semantic representation by qualia roles to be defeasible. So, *He particularly enjoyed your book* is interpreted to mean *He enjoyed doing something to your book, which is reading if nothing speaks against it.* Since in this example the discourse structure as well as the knowledge that goats usually don't read do speak against this assumption it is overridden.

3.4.3 Comments

Copestake and Lascarides (1998) introduce a considerable amount of flexibility into Pustejovsky's analysis of polysemy and metonymy. This makes possible a treatment of phenomena which cannot be explained strictly lexically. However, still not all cases of metonymy are captured. For instance, *I am parked out back* can be treated neither by constructional polysemy nor by sense extension. Simply applying a lexical rule is not possible, because of the acceptability of *I am parked out back and have been waiting for 15 minutes.* It also seems implausible to include reference to a vehicle in the qualia structure of a person, which would be necessary for a treatment by constructional polysemy. This means that semantic construction would probably fail due to a sort conflict. It seems that a more liberal interaction of pragmatics must be allowed.

3.5 Summary

This chapter has presented several accounts of meaning shift which rely on different sources of information (lexical, sortal, pragmatic) to find an appropriate shift operator. We briefly presented the main mechanisms which allow for the addition of a shift operator to the semantic representation and which guide the search for an appropriate operator in a given context. The

discussion in this and the preceding chapter made clear that any type of knowledge and even an interaction between different types may play a role in determining the shift operator. These different types of information are not necessarily available at the same stage of linguistic processing, so that it is not possible to restrict operations of meaning shift to e.g. the semantic construction process. This calls for an approach which provides a representation which is flexible enough to allow for different types of information to contribute to the semantic representation at any time in the course of linguistic processing. Copestake and Lascarides (1998) try to account for the influence pragmatics may have by treating certain information as defeasible default information which can be overridden by pragmatic constraints. Hobbs et al. (1993) and Dölling (1997) delay the decision on what material to add until all information necessary is available and can be evaluated simultaneously by introducing underspecification at all places where the addition of a shift operator is possible. As already mentioned several times, this is very similar to Egg's (1999) approach. He proposes a constraint based underspecified description of meaning shifts which can be made more specific at any time by simply adding further constraints.

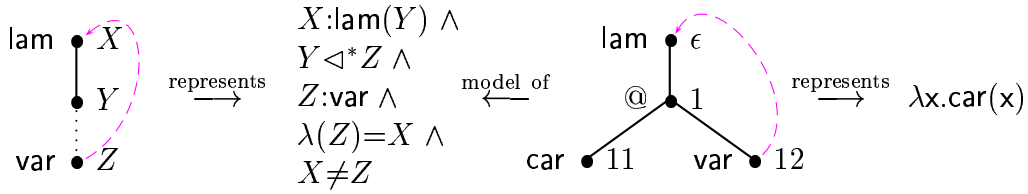
Chapter 4

Semantic Underspecification With Tree Descriptions

In this chapter we will define the underspecification formalism that we will work with in the rest of this thesis. It exploits the idea that logical formulas, in particular λ-terms, can be viewed as trees. These trees constitute the object language of our underspecification formalism. As meta language we use a constraint language called *Constraint Language for Lambda Structures* (CLLS) (Koller 1999; Egg et al. 1998) which allows us to describe sets of trees and thus allows for an underspecified description of logical formulas. The main underspecification instrument are dominance constraints.

Figure 4.1 illustrates the levels of representation that are important for CLLS. As said above, CLLS is a constraint language. CLLS constraints describe λ-structures. λ-structures are tree-like and represent λ-terms. We will talk about λ-structures in Section 4.1. CLLS constraints can be represented by constraint graphs. In fact, we will normally present constraints as graphs throughout this thesis, because it makes them much easier to read. The language of CLLS and the relation between constraints and the graphical representation will be discussed in Section 4.2.

Finally, we will see some example applications and a syntax/semantics inter-



(a) constraint graph (b) constraint (c) λ-structure (d) λ-term

Figure 4.1: Levels of Representation in CLLS

face which derives underspecified descriptions in the presented formalism in Sections 4.3 and 4.4 respectively.

4.1 λ-terms as Trees: the Object Language

λ-terms are widely used for representing natural language semantics. They can be represented by tree-like structures called λ-structures. A λ-term and its corresponding λ-structure are given in Figure 4.2.

Figure 4.2: A λ-term and the corresponding λ-structure

Variable binding and functional application are made explicit in the λ-structure by the dashed arrow and the label @, respectively. Linking variables to their binders explicitly instead of via names results in a stable binding relation, which can never be undone (accidentally). So, a λ-structure corresponds to a unique class of λ-terms modulo α-equality.

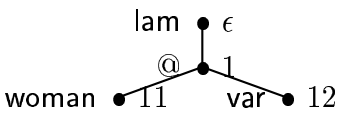
Since λ-structures are very close to trees, we will first define tree structures and then extend this definition.

4.1.1 Tree Structures

A tree structure consists of a tree domain, which defines the nodes and their position in the tree, and a labeling function, which assigns a label to every node.

A node is a word $\pi \in \mathbb{N}^*$, where $\mathbb{N}$ is the set of positive integers and $\mathbb{N}^*$ is the set of words over $\mathbb{N}$. 1, 123 or 111, for instance, may be nodes in a tree. We write ϵ for the empty word, corresponding to the root node. Intuitively, a tree can now be described by a set of nodes as follows: ϵ is the root node, 1 is the first, the left most, daughter of the root, 2 is the second daughter of the root, 21 is the left most daughter of the second daughter of the root, and so forth. We write $\pi\pi'$ for the concatenation of $\pi \in \mathbb{N}^*$ and $\pi' \in \mathbb{N}^*$ and we say that π is a *prefix* of π' if there is a word π'' such that $\pi\pi'' = \pi'$. πi is the *left sibling* of πj iff $j = i + n$, for $1 \leq n \leq j - 1$.

A *tree domain* Δ is a nonempty subset of $\mathbb{N}^*$, which is closed under the prefix and the left-sibling relations. Figure 4.3 gives an example of a tree structure and its tree domain.



$$\Delta = \{\epsilon, 1, 11, 12\}$$

Figure 4.3: A Tree Structure and its Tree Domain

When talking about trees, we will usually say that a node π *dominates* a node π' ($\pi \lhd^* \pi'$) iff π is a prefix of π'. Furthermore, we will say that nodes π_1 and π_2 are *disjoint with branching point* π_0, written as

$$\pi_1 \perp \pi_2 \text{ at } \pi_0,$$

iff there are words π_1' and π_2' and different positive integers i, k such that $\pi_1 = \pi_0 i \pi_1'$ and $\pi_2 = \pi_0 k \pi_2'$. In Figure 4.3, for example, 11 and 12 are disjoint with branching point 1, $11 \perp 12$ at 1.

Furthermore, we assume a signature $\Sigma = \{@, \mathsf{lam}, \mathsf{var}, \mathsf{woman}, \dots\}$ of node labels. Each of these labels is assigned a fixed arity by an arity function $\mathsf{ar} : \Sigma \to \mathbb{N}_0$, where $\mathbb{N}_0$ is the set of nonnegative integers. This arity determines the number of daughters a node with a given label has. For example, $\mathsf{ar}(@) = 2$.

Definition 1 (Tree Structure). A *tree structure* is a tuple (Δ, σ) consisting of a finite tree domain Δ and a total *labeling function* $\sigma : \Delta \to \Sigma$ such that for all $\pi \in \Delta$ and $i \in \mathbb{N}$:

$$\pi i \in \Delta \;\Leftrightarrow\; 1 \leq i \leq \mathsf{ar}(\sigma(\pi)).$$

The labeling function of the tree structure in Figure 4.3 can be given by $\sigma(\epsilon) = lam, \sigma(1) = @, \sigma(11) = woman, \sigma(12) = var$.

4.1.2 λ-Structures

A λ-structure is a tree structure extended with a partial λ binding function.

Definition 2 (λ-Structure). A *λ-structure* is a triple $(\Delta, \sigma, \lambda)$ consisting of a tree structure (Δ, σ) and a partial *λ-binding function* $\lambda : \Delta \rightsquigarrow \Delta$ which satisfies for all $\pi, \pi' \in \Delta$:

$$\text{if } \lambda(\pi) = \pi' \text{ then } \begin{cases} \sigma(\pi) = \mathsf{var},\ \sigma(\pi') = \mathsf{lam}, \\ \text{and } \pi' \lhd^* \pi \end{cases}$$

So the λ-structure in Figure 4.2 its tree domain, $\{\epsilon, 1, 11, 12\}$, the λ-binding function, given by $\lambda(12) = \epsilon$, and the labeling function, $\sigma(\epsilon) = \mathsf{lam}$, $\sigma(1) = @$, $\sigma(11) = \mathsf{woman}$, and $\sigma(12) = \mathsf{var}$.

Note that this definition allows for free variables, which is important for some of the proofs in the next chapter to work out.

4.2 Graphs as Tree Descriptions: the Meta-Language

We will now define a fragment of CLLS, the Constraint Language for Lambda Structures. CLLS is a language that describes sets of λ-structures by stating constraints on the nodes of the tree domain. The fragment of CLLS we are interested in consists of labeling, dominance, λ-binding, and inequality constraints. Full CLLS additionally provides parallelism constraints, for modeling ellipsis, and a linking relation between nodes, which makes anaphoric

binding explicit. Full CLLS allows for an integrated treatment of the interaction of scope ambiguity, ellipsis, and anaphora (Egg et al. 1998; Xu 1998).

We will start by defining the syntax and semantics of CLLS. Afterwards we will introduce some further concepts which will become important later.

4.2.1 Syntax and Semantics of CLLS

Assuming an infinite set of (node) variables ranged over by X, Y, Z, the syntax of the fragment of CLLS we are going to use is the following:

$$\begin{array}{rcll} \varphi & ::= & X{:}f(X_1, \ldots, X_n) & (f \in \Sigma \text{ and } \mathsf{ar}(f) = n) \\ & | & X \lhd^* Y & \\ & | & \lambda(X){=}Y & \\ & | & X{\neq}Y & \\ & | & X \perp Y \text{ at } \{Z_1, \ldots, Z_n\} & \\ & | & \varphi \wedge \varphi'. & \end{array}$$

We will call the set of variables of a constraint φ $\mathcal{V}(\varphi)$ and we will say that $\varphi' in \varphi$ if all atomic constraints of φ' are also contained in φ.

A *variable assignment into a λ-structure* $\mathcal{M}$ is a total function from the set of variables to the domain of $\mathcal{M}$. A pair $(\mathcal{M}, \alpha)$ of a λ-structure $\mathcal{M}$ and a variable assignment α into $\mathcal{M}$ *satisfies* a constraint φ iff it satisfies all of its atomic constraints in the following way:

$X{:}f(X_1, \ldots, X_n)$ is satisfied iff $\sigma(\alpha(X)) = f$ and $\alpha(X)i = \alpha(X_i)$ for $1 \leq i \leq n$.

$X \lhd^* Y$ is satisfied iff $\alpha(X) \lhd^* \alpha(Y)$.

$\lambda(X) = Y$ is satisfied iff $\lambda(\alpha(X)) = \alpha(Y)$.

$X{\neq}Y$ is satisfied iff $\alpha(X){\neq}\alpha(Y)$.

$X \perp Y$ at $\{Z_1, \ldots, Z_n\}$ is satisfied iff there is an i, $1 \leq i \leq n$, such that $\alpha(X) \perp \alpha(Y)$ at $\alpha(Z_i)$.

The generalized version of disjointness with a branching point we have defined here will be handy in some of the proofs of the next section. If the branching point is not important we will also write $X \perp Y$, meaning that there is a π such that $\alpha(X) \perp \alpha(Y)$ at π holds.

A pair $(\mathcal{M}, \alpha)$ which satisfies a constraint φ is also called a *solution* of φ, and $\mathcal{M}$ a *model* of φ. We write $\varphi \models \varphi'$ and say that φ *entails* φ' if every solution of φ is a solution of φ'. The notions of solutions and entailment can be lifted to first-order formulas built from constraints as usual.

For illustration, have a look at Figure 4.4, which presents two only slightly different (lightgray area) CLLS constraints and some λ-structures that satisfy them.

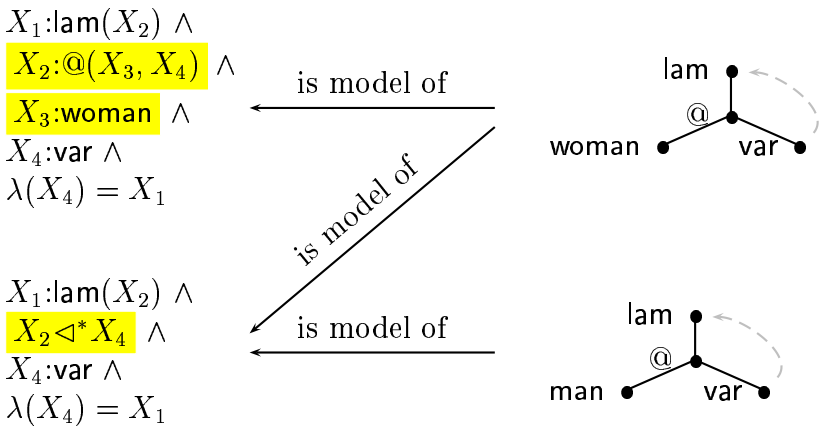


Figure 4.4: Constraints Describe Sets of λ-Structures

We will call a solution *constructive* if for every node π in the domain of $\mathcal{M}$ there exists a variable $X \in \mathcal{V}(\varphi)$ such that $\alpha(X) = \pi$ and $X{:}f(\dots)$ *in* φ for some label f in Σ. This means that constructive solutions are built only from material present in the description. Every labeled node in the solution must correspond to a labeling constraint in φ and every node must be denoted by an element of $\mathcal{V}(\varphi)$. The model given for the upper constraint in Figure 4.4 is a constructive solution, while the solutions given for the lower constraint are not. For linguistic applications, we are usually interested in constructive solutions for the constraints derived by semantic construction, e.g. when resolving scope ambiguities. There are some phenomena, however, which

cannot be modeled when assuming only constructive solutions. Meaning shift is one of them, since it may require new material to be added to the semantic representation derived by semantic construction.

4.2.2 Normal Constraints and Constraint Graphs

Another restriction which is reasonable to make for linguistic applications is that all constraints we have to deal with are *normal*, i.e. two labeled variables must denote distinct nodes:

Definition 3 (Normal Constraints). A constraint φ is *normal* if for each two labeling constraints $X{:}f(X_1, \ldots, X_n)$ *in* φ and $Y{:}g(Y_1, \ldots, Y_m)$ *in* φ, where X and Y are different variables, it holds that $X{\neq}Y$ *in* φ (even if $f = g$).

Normality ensures that semantic material introduced by different occurrences of the same substring in the utterance will never be unified. For instance, in the sentence *A cat chases a cat* both occurrences of *a cat* introduce the same material to the semantic representation. Without the restriction to normality these two subconstraints can be mapped to the same part of a solution, thereby yielding a semantic representation corresponding to *A cat chases itself*, which clearly is something different.

Since constraints easily become very hard to read, we will often draw them as graphs. Nodes in these graphs stand for variables in the constraint, which are variables for nodes of a tree. They should not be confused with nodes in λ-structures, which are nodes of a tree structure. In a solution, two different nodes of a constraint graph, which belongs to the meta-language level of representation, may be mapped to just one node in a λ-structure, which is at the object-language level of representation. The constraints in Figure 4.4, for instance, can be represented by the graphs in Figure 4.5.

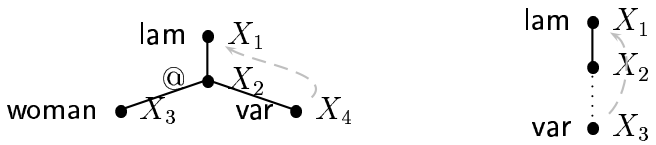


Figure 4.5: Constraint Graphs

We will take graphs to always represent normal constraints, which means that rigid fragments, i.e. parts which are connected by solid edges only, may not overlap.

4.2.3 Fragments

This takes us to the next useful concept: *fragments*. A fragment in a constraint graph φ is a subset of the variables of φ such that they are connected by labeling constraints.

Definition 4 (Fragments). Let φ be a normal CLLS constraint.

1. Let two variables $X, Y \in \mathcal{V}(\varphi)$ be *connected* if $X{:}f(\dots Y \dots)$ *in* φ or if there is a third variable $Z \in \mathcal{V}(\varphi)$ such that X and Z are connected and $Z{:}f(\dots Y \dots)$ *in* φ.

2. A *(rigid) fragment* of φ is a subset $F \subseteq \mathcal{V}(\varphi)$ of variables that are pairwise connected in φ. A node $X \in F$ is called a *leaf* of a fragment F if F contains no variable Y such that there is a labeling constraint $X{:}f(\dots Y \dots)$ in φ. A *hole* is a leaf which is not labeled at all.

Fragments are "tree-like". Every fragment F has the following properties:

1. For each two variable in the fragment it holds that either one dominates the other or they are disjoint (cf. Lemma 5).

2. F has a unique *root*, i.e. contains a variable that must dominate all other members of F in any solution (cf. Lemma 6).

3. In particular, all leaves of F must always denote pairwise disjoint nodes (cf. Lemma 7).

Lemma 5. *Let φ be a normal constraint, let F be a fragment in φ, let X, Y be different variables in F, and let $(\mathcal{M}, \alpha)$ be a solution of φ. Then there is a $W \in F$ such that the following entailment holds:*

$$\varphi \models X \lhd^* Y \vee Y \lhd^* X \vee X \perp Y \text{ at } \{W\}.$$

Lemma 6. *Let φ be a normal constraint, let F be a fragment in φ, and let $(\mathcal{M}, \alpha)$ be a solution of φ. Then F contains exactly one variable R such that*

$$\varphi \models \bigwedge_{X \in F} R \lhd^* X.$$

Lemma 7. *Let φ be a normal constraint, let F be a fragment in φ, and let X, Y be different leaves of F. Then there is a variable $W \in F$ such that*

$$\varphi \models X \perp Y \text{ at } \{W\}.$$

The proofs of Lemmas 5 – 7 can be found in Section A.1 of the appendix.

4.3 Application to Scope Ambiguity

In the introduction we saw an example of scope ambiguity, which is repeated here as Example 4.1.

(4.1) Every man loves a woman.

We will now demonstrate by means of this example, what an underspecified treatment of scope ambiguities looks like in CLLS. The λ-structures corresponding to the two readings of 4.1 are displayed on the left hand side of Figure 4.6. A constraint graph describing exactly these two λ-structures (when considering only constructive solutions) is given on the right.

We can distinguish three rigid fragments in the constraint graph, namely the two quantifiers and the verbal nucleus. To make it easier to talk about these fragments, we have marked them with Roman numbers. Both of the fragments corresponding to the quantifiers, fragments I and II, have a hole which dominates the root of the third fragment, but the constraint contains no information on the position of these two fragments with respect to each other. This is what accounts for the scope ambiguity. In finding (constructive) solutions for this constraint, the given fragments have to be arranged into a tree. Since upward branching is not allowed in trees and since in a normal constraint overlapping of fragments (except of holes with roots) is not allowed, the only way to achieve this is to identify node X_3 with node X_5 and node X_4 with X_1, or node X_4 with node X_5 and node X_3 with X_2. This yields exactly the two scope readings of this sentence.

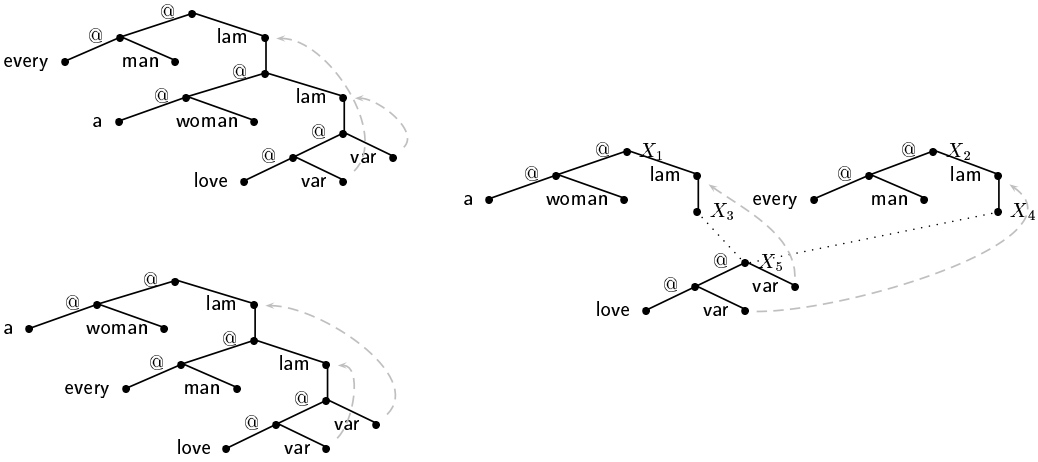


Figure 4.6: Scope Ambiguities in CLLS

4.4 Semantic Construction

Now, we will present a syntax/semantics interface which derives underspecified semantic representations in the form of CLLS constraints.

For purposes of the presentation, we will consider only a small grammar fragment here. But it should be a simple matter to extend it to cover other phenomena as well. In fact, we have implemented an HPSG grammar which produces essentially the same constraints and has a much wider coverage. The grammar fragment is given in Figure 4.7 by a set of simple context-free grammar rules (where IV stands for intransitive verb, TV for transitive verb, CV for control verb, and *Lex* is a relation between words W and lexical categories $\alpha \in \{\texttt{Det}, \texttt{N}, \texttt{IV}, \texttt{TV}, \texttt{CV}, \texttt{Adv}\}$ which represents the lexicon).

The syntax/semantics interface of the grammar associates subconstraints with each node ν of the parse tree. The contributions of these are then conjoined. The rules by which subconstraints are introduced are presented in Figure 4.8. We take $[_{\nu:P}\ Q\ R]$ to mean that node ν in the syntax tree is labeled with P, and its two daughter nodes $\nu 1$ and $\nu 2$ are labeled with Q and R, respectively. (Remember that we represent trees, also the parse tree, by a tree domain $\Delta \subseteq \mathbb{N}^*$ and a labeling function $\sigma{:}\Delta \to \Sigma$; cf. Section

(a1) $\texttt{S} \rightarrow \texttt{NP VP}$

(a2) $\texttt{NP} \rightarrow \texttt{Det } \overline{\texttt{N}}$

(a3) $\overline{\texttt{N}} \rightarrow \texttt{N}$

(a4) $\overline{\texttt{N}} \rightarrow \texttt{N PP}$

(a5) $\texttt{PP} \rightarrow \texttt{P NP}$

(a6) $\texttt{VP} \rightarrow \texttt{VP Adv}$

(a7) $\texttt{VP} \rightarrow \texttt{IV}$

(a8) $\texttt{VP} \rightarrow \texttt{TV NP}$

(a9) $\texttt{VP} \rightarrow \texttt{CV VP}$

(a10) $\alpha \rightarrow W$ if $(W, \alpha) \in Lex$

Figure 4.7: The Grammar

4.1.1.Hence, $\nu, \nu 1, \nu 2 \in \mathbb{N}^*$) One variable of the subconstraint associated with node ν is specially distinguished. We will call it X_ν. X_ν is related to $X_{\nu 1}$ and $X_{\nu 2}$ via the CLLS constraint.

Some other nodes may also be given a special name, e.g. X_ν^{scope} in the contribution of rule (b1). This is to make it easier to refer to them later on. Furthermore, it clarifies their function in the final constraint; in the example, X_ν^{scope} is intuitively the scope of the quantifier represented by the NP.

Figure 4.8 doesn't completely specify all the necessary λ-binding constraints. This is because variable binding requires words to have access to information about their arguments. Within a a context free grammar this information is difficult to maintain, but in any "serious" grammar formalism (HPSG, LFG) it doesn't pose a problem. For the rules (b7), (b8), and (b9), we therefore assume the availability of additional information about the subjects of the verbs considered. Suppose that ν is a VP node in the parse tree and ν' is the NP node that represents the subject, then we add the following λ-binding constraint:

$$\lambda(X_\nu^{arg1}) = X_{\nu'}^{scope}$$

Similarly for rule (b5), if ν' is the NP node modified by the PP at ν then we add the following λ-binding constraint:

$$\lambda(X_\nu^{arg1}) = X_{\nu'}^{restr}$$

Now, let's look at how semantic construction works. Figure 4.9 gives the

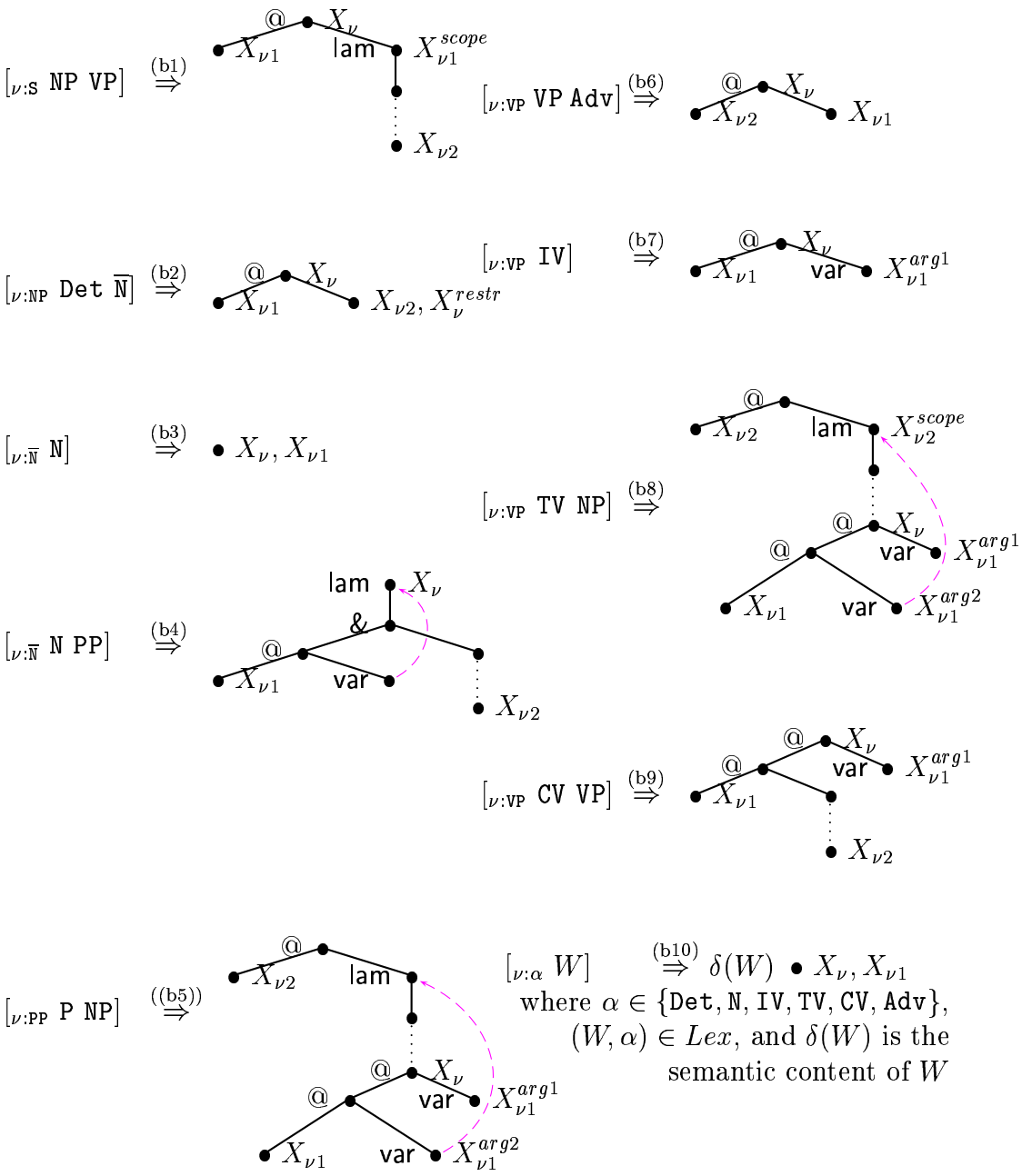


Figure 4.8: The syntax/semantics interface

parse tree for our well-known example sentence *Every man loves a woman* on the left hand side and the semantic representation derived by our syntax/semantics interface on the right hand side. The dotted arrows go from a node ν in the parse tree to node X_ν in the constraint graph, indicating which part of the constraint is the semantic contribution of node ν. The dotted arrows are labeled with the rules that were used.

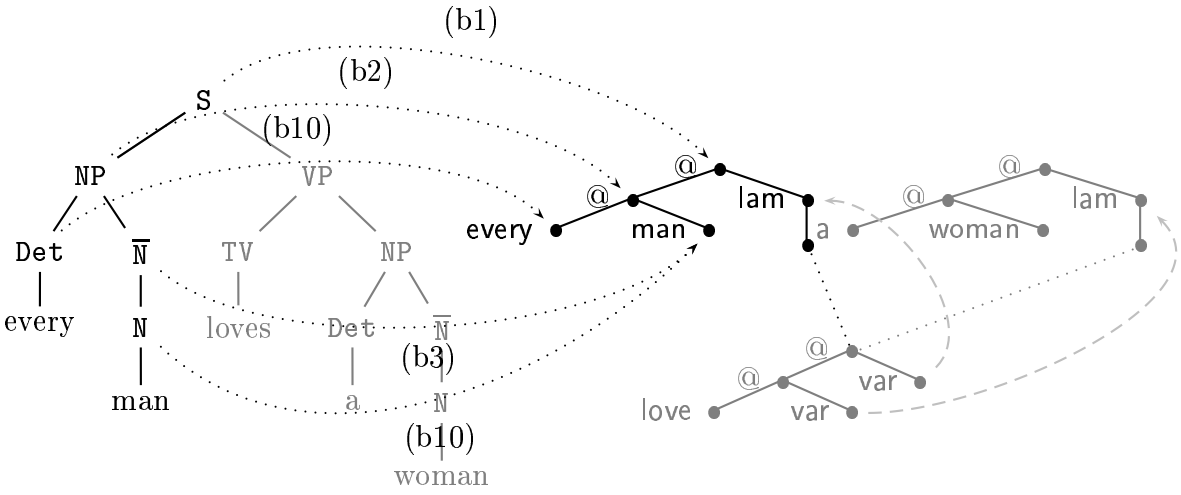


Figure 4.9: *Every man loves a woman*: Semantic Construction

4.5 Summary

This chapter has presented the Constraint Language for Lambda Structures (CLLS), which is a tree description language based on dominance constraints. There are four levels of representation which are important when working with CLLS: *Constraints* describe sets of *λ-structures*. λ-structures represent *λ-terms* and constraints can be represented as *constraint graphs*. We demonstrated the application of CLLS to linguistic phenomena of semantic ambiguity. We furthermore developed a syntax/semantics interface which derives underspecified semantic representations in the form of CLLS constraints.

Chapter 5

Modeling Meaning Shifts in an Underspecification Framework

In this chapter, we will motivate and present Egg's (1999) account of meaning shifts. By making use of the underspecification techniques introduced in the previous chapter, he can provide a monotonic treatment of meaning shifts. We will adapt the syntax/semantics interface given in the last chapter to his account. We will see that the general strategy Egg is using can lead to overgeneration. However, in the application to meaning shifts it never does. We will prove this after having developed some techniques for drawing inferences about dominance and disjointness which powerfully support the type of inferences we are interested in.

The proof of the main result (Theorem 18) will actually be given in this chapter. For all other proofs, we refer the reader to the appendix.

5.1 Relaxing Underspecified Semantic Representations

Recall from the introduction that we want to model meaning shifts by a process which adds information, the shifting operator, to a semantic representation. In terms of tree descriptions as semantic representations, the problem presents itself as depicted in Figure 5.1. On the left hand, we have the semantic representation the syntax/semantics interface presented in the previous chapter would derive for the sentence *I am parked out back.* On the right hand, we have a representation of the intended meaning. The shift operator was inserted into the semantic representation by some process of tree adjunction at node X.

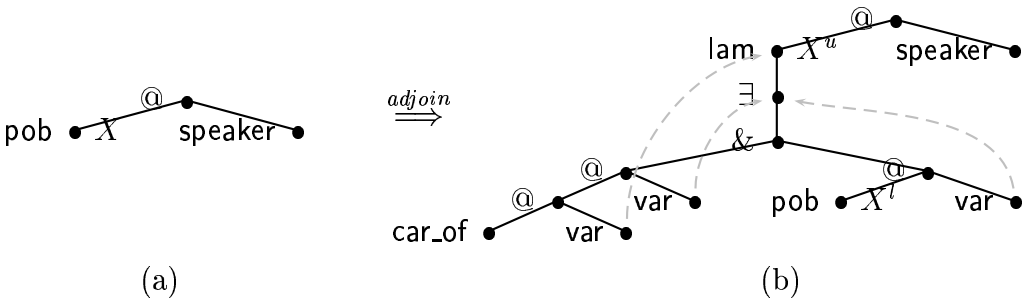


Figure 5.1: Meaning Shift by Tree Adjunction

We saw in Chapter 2 that which information is added and if information is added depends on lexical as well as contextual factors. Lexical information can be accessed during the process of semantic construction, but context, world, or pragmatic knowledge is usually not yet available at this stage. This implies that a final decision on a specific shift operator can only be made after semantic construction, when all of the sources of information needed can be accessed. So, phenomena of meaning shift can be seen as cases of incomplete information, i.e. not all of the information needed is provided by purely linguistic knowledge. With the perspective on underspecification put forward in Section 1.2 of the introduction this suggests a treatment by underspecification, which is what Egg proposes.

In Egg's framework semantic construction derives an underspecified representation which contains no sort conflict and describes all potential meaning

shifts as well as the basic readings involving no meaning shifts. The key idea is to have semantic construction derive semantic representations which are *relaxed*, i.e. made less specific by introducing an additional dominance edge, at the site where the shift operator is to be introduced. The semantic representation of *I am parked out back* will look like displayed in Figure 5.2 (a).

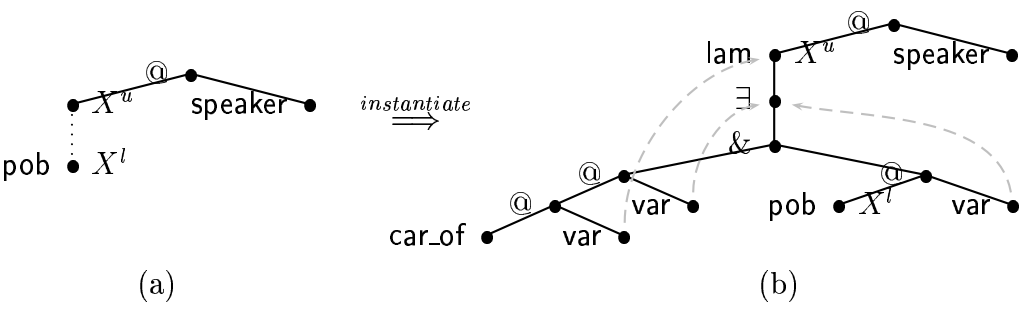
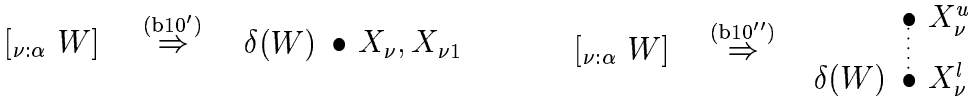


Figure 5.2: Meaning Shift by Instantiation of Gaps

The sort conflict contained in the unrelaxed version is cushioned here by the additional dominance edge. We will call this dominance edge *relaxation gap* since it functions like a gap in the semantic representation by allowing for material to go between the nodes X^u and X^l. In particular, the shift operator can be go into the relaxation gap. Hence, by this method the actual process of meaning shift boils down to instantiation of gaps. We will call X^u and X^l the *upper* and the *lower boundary* of the relaxation gap, respectively.

We saw in Chapter 2 that, in case of the phenomena of meaning shift we are interested in here, it is usually the predicate introduced by the verb to which the shift operator is introduced. For the time being, we will only consider this type of meaning shift. As said above, we want semantic construction to derive only relaxed semantic representations. In order to achieve this, the process of relaxation is compiled into the syntax/semantics interface, which means that rule (b10) of the interface presented in the previous chapter is replaced by rules (b10') and (b10") shown in Figure 5.3.

The syntax/semantics interface controls, where the potential application sites for shift operators are, and it derives **only** constraints which are relaxed at these positions. In case no meaning shift is necessary, the gap can just be closed; otherwise it provides space for introducing new material, like shift



where:
$\alpha \in \{\texttt{Det}, \texttt{N}, \texttt{Adv}\}$, $(W, \alpha) \in Lex$,
and $\delta(W)$ is the semantic content of W

where:
$\alpha \in \{\texttt{IV}, \texttt{TV}, \texttt{CV}\}$, $(W, \alpha) \in Lex$,
and $\delta(W)$ is the semantic content of W

Figure 5.3: Revised Syntax/Semantics Interface

operators. Note, that this adding of new material is a non-destructive operation; new atomic constraints are simply added to the semantic representation. This is where Egg's approach differs from an approach to meaning shift by tree adjunction.

5.1.1 An Integrated Treatment of Meaning Shifts and Scope Ambiguity

Using general underspecification techniques to model meaning shifts makes an integration into a general underspecification formalism suggest itself. And in fact, it is possible to describe sentences involving scope ambiguities in addition to meaning shifts, as in the following example.

(5.1) Every driver of a mafia boss is parked out back.

The corresponding semantic representation as derived by the revised version of our syntax/semantics interface is given in Figure 5.4.

In reading the constraint graph in Figure 5.4, it is first of all helpful to identify its various fragments. For convenience, I have labeled them with Roman numbers.

The way in which fragments *I*, *II*, and *III* are connected accounts for the scope ambiguity between the two quantifiers *a mafia boss* (represented by fragment *I*) and *every driver* (represented by fragment *II*). We have expressed that they both have to outscope the semantic contribution of *of*

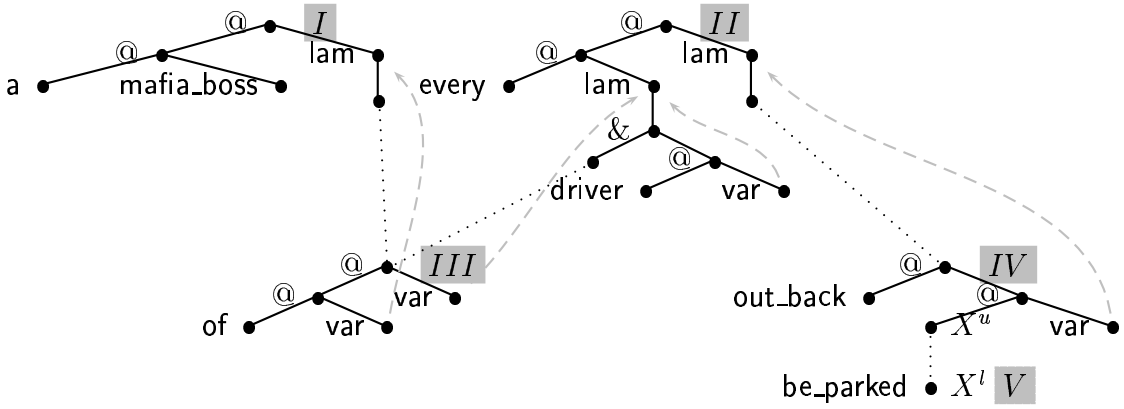


Figure 5.4: Underspecified Representation of a Potential Meaning Shift

(fragment *III*), but we haven't said in what relative position they are to each other. A meaning shift has to account for the fact that the sentence is understood to mean that the vehicles of the mafia bosses are parked out back and not the persons themselves. This can be modeled by filling an appropriate shift operator into the gap left open by the dominance edge $X^u \triangleleft^* X^l$ in the description of the verb semantics.

5.1.2 A Problem?

Locally, the relaxation site in Figure 5.4 looks just the same as in the simple example above (Figure 5.2 (a)). And intuitively, it should be treated in the same way. This would mean that adding the shift operator was possible before enumeration of scope readings and that these phenomena weren't affected at all by their co-occurrence. But is it always safe to just ignore other ambiguities while adding the shift operator? The following discussion will show that we really flushed a potential problem here.

The rather abstract example given in Figure 5.5 illustrates the problematic effect. 5.5(b) is a result of relaxing 5.5(a) and 5.5(c) is a possible solution of 5.5(b). Observe that in this solution the fragment $f(\bullet)$ has slipped into the

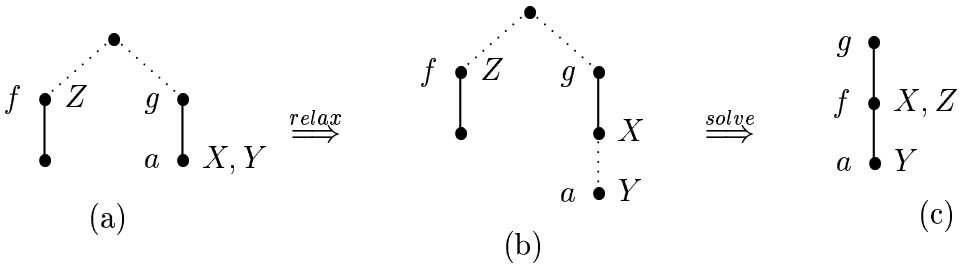


Figure 5.5: Problem

newly opened gap. The relaxed constraint has a solution which differs from the solutions for its unrelaxed counterpart not only at the relaxation site, but in the general structure, i.e. the fragments of the relaxed constraint can be arranged in a way which is not possible in the case of its unrelaxed counterpart. The solution in Figure 5.5(c) is possible because of an interaction of the relaxation site with other dominance edges, which, as argued above, should never happen.

A closer look at the dominance edges in our graphs will help us understand where the problem stems from. We notice that we have two kinds of dominance edges in our graph which serve fundamentally different purposes. In the case of scope ambiguities, the gaps are only there so that the material already present in the semantic representation can be arranged into tree structures in different ways. In the case of meaning shifts on the other hand, we introduced them to make room for the shift operator. Material totally new to the semantic representation, and only this, is to be filled in. In Figure 5.5 the "purpose" of the relaxation gap was violated since old material, i.e. material already present in the semantic representation, was filled into the gap. Figure 5.5 is therefore called an *unintended solution.* Solutions where no old material has slipped into the relaxation site are called *intended solutions.*

The constraint language does not give us enough expressive power to prevent old material to slip into relaxation gaps, because it doesn't provide negation or disjunction of constraints. This doesn't matter, though, since there actually is a much better way out of this problem. We can prove that our revised

syntax/semantics interface never gives rise to the problematic effect described above. The structure of any constraint generated by this syntax/semantics interface is such that all of its solutions are intended; no old material can ever slip into any of the relaxation gaps. The advantage is that this proof only has to be done once. Then we never have to worry about the problem again and can safely relax the semantic representations to model meaning shift. We will work out this proof in the rest of the chapter.

5.2 Safety of Relaxation

The goal of this section is to find a general criterion which tells us for a certain relaxation of a semantic representation whether it can happen that old material slips into the relaxation gap. We will start by defining *intended solutions*. Then we will state a *Safety Criterion* which, if satisfied by a constraint, ensures that all solutions of that constraint are intended. We will introduce *chains of fragments* which are subconstraints with a certain structure that powerfully support the proof of satisfiedness of the Safety Criterion. Finally, we will prove that all constraints derived by our syntax/semantics interface satisfy the Safety Criterion.

5.2.1 Intended Solutions

As said in the previous section, intended solutions are solutions of the constraint in which there is no old material at the position where the gap was. Consider Figure 5.6 for illustration. The constraint graph to the left gets relaxed at node X: node X is replaced by two new nodes X^u and X^l which are connected by a dominance edge. This introduces the relaxation gap. On the right is a solution of the relaxed constraint. The gap was filled, so that there is now material which is dominated by X^u, but not by X^l. Note that in the example some of this material is old material, node Y for instance. Hence, this is an unintended solution.

Now, how can we tell that old material has slipped into the gap? Intuitively, all solutions of relaxed graphs should differ from solutions of their unrelaxed counterparts only at the relaxation site. That is, if we cut out that part of the solution where the relaxation gap used to be (namely everything dominated

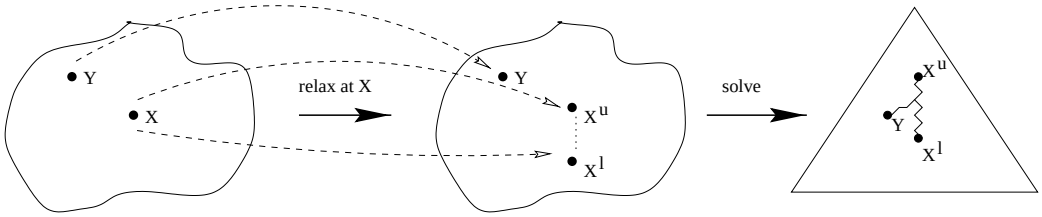


Figure 5.6: Old Material Slipped into a gap

by X^u but not by X^l), then we should arrive at a solution for the unrelaxed graph. However, if old material is in this area, as it is in Figure 5.6, it will disappear. Hence the result of this cutting out operation, which we will call *projection* cannot be a solution of the unrelaxed graph, because material present in the constraint is missing in this resulting tree.

So, intended solutions of a relaxed graph are those solutions of this graph which yield a solution of its unrelaxed counterpart when the material which was filled into the relaxation gap is cut out. This relationship between constraints, their relaxations, and their solutions is depicted by Figure 5.7.

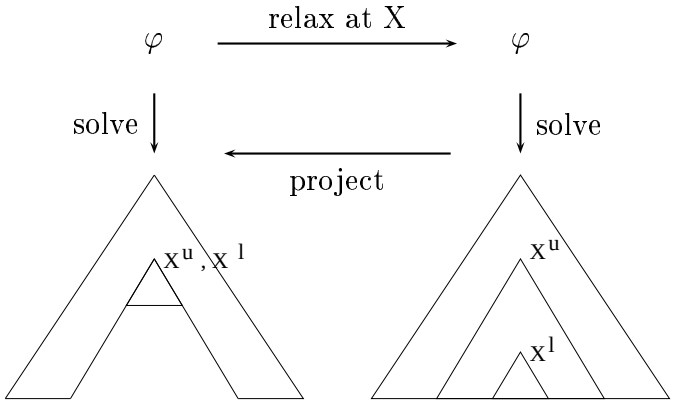


Figure 5.7: Intended Solutions

In order to arrive at a formal definition of intended solution we have to define the operations of relaxation and projection.

Let's begin with the (formal) definition of relaxation of a constraint at a certain node X. The idea is that X is split in two: an *upper variable* X^u and a *lower variable* X^l. Every atomic constraint that "refers to X from above" refers to X^u in the relaxed constraint and every atomic constraint that "refers to X from below" refers to X^l. The atomic constraints that don't refer to X are not affected. We will now make more precise what "referring to X from above/below" means.

Let φ be a normal constraint, $X \in \mathcal{V}(\varphi)$ a variable, and $\mathsf{eq}_\varphi(X) = \{Y \in \mathcal{V}(\varphi) \mid \varphi \models X \lhd^* Y \wedge Y \lhd^* X\}$ the set of variables which must be equal to X. In a satisfiable normal constraint, we can compute $\mathsf{eq}_\varphi(X)$ syntactically by reflexivity and transitivity inferences about dominance constraints.

We partition the set of atomic constraints in φ into three parts, depending on how they refer to $\mathsf{eq}_\varphi(X)$. The set $U_X(\varphi)$ contains all atomic constraints in φ that relate a $Z \in \mathsf{eq}_\varphi(X)$ to a variable "above", i.e. that are of one of the following forms (for some $f, Y, n \geq 1, Z_1, \ldots, Z_n$):

$$\begin{gathered} Y{:}f(\ldots, Z, \ldots),\ Y \lhd^* Z, \\ Y \perp Z \text{ at } \{Z_1, \ldots, \mathcal{Z}_n\} Z{\neq}Y, \text{ or } Y{\neq}Z \end{gathered}$$

The set $L_X(\varphi)$ contains all atomic constraints in φ that relate a $Z \in \mathsf{eq}_\varphi(X)$ to some variable "below", i.e. which are of the forms:

$$\begin{gathered} Z{:}f(\ldots, Y, \ldots),\ Z \lhd^* Y, \\ \lambda(Y){=}Z, \lambda(Z){=}Y, \\ Z{\neq}Y, \text{ or } Y{\neq}Z \end{gathered}$$

The set $O_X(\varphi)$ contains those atomic constraints of φ in which no variable of $\mathsf{eq}_\varphi(X)$ occurs.

In the following definition, we write $[Y/\mathcal{V}]$ for the substitution that maps all variables in the variable set $\mathcal{V}$ to the variable Y.

Definition 8 (Relaxation). Let φ be a normal constraint, X a variable in $\mathcal{V}(\varphi)$ such that a labeling constraint $X{:}f(\ldots)$ *in* φ, and X^u, X^l variables fresh for φ. Then the *relaxation* $\mathcal{R}_X(\varphi)$ of a constraint φ at the variable X is defined as the conjunction

$$\begin{aligned} X^u \lhd^* X^l \quad & \wedge \bigwedge\{\mu[X^u/\mathsf{eq}_\varphi(X)] \mid \mu \in U_X(\varphi)\} \\ & \wedge \bigwedge\{\mu[X^l/\mathsf{eq}_\varphi(X)] \mid \mu \in L_X(\varphi)\} \\ & \wedge \bigwedge\{\mu \mid \mu \in O_X(\varphi)\}. \end{aligned}$$

Lemma 9. *The relaxation $\mathcal{R}_X(\varphi)$ of a normal constraint φ at X is normal.*

Next, we will define the projection operation. This is the operation that "cuts" a part out of a λ-structure.

Definition 10 (Projection). Let $\mathcal{M}$ be a λ-structure $(\Delta, \sigma, \lambda)$, and let $\pi_u \lhd^* \pi_l$ be nodes in Δ. Let $\Delta^{\pi_u}_{\pi_l} \subseteq \mathbb{N}^*$ be the subset of Δ without the tree fragment between π_u and π_l:

$$\Delta^{\pi_u}_{\pi_l} = \{\pi \in \Delta \mid \text{ if } \pi_u \lhd^* \pi \text{ then } \pi_l \lhd^* \pi\}$$

The *projection* $p : \Delta^{\pi_u}_{\pi_l} \to \mathbb{N}^*$ is the function which satisfies for all paths $\pi \in \Delta^{\pi_u}_{\pi_l}$:

$$p(\pi) = \begin{cases} \pi & \text{if not } \pi_u \lhd^* \pi \\ \pi_u \pi' & \text{if } \pi = \pi_l \pi' \end{cases}$$

Furthermore, note that p is one-to-one and that $p(\Delta^{\pi_u}_{\pi_l})$ is a tree domain. The projection $\mathcal{M}^{\pi_u}_{\pi_l}$ of the λ-structure $\mathcal{M}$ at nodes π_u, π_l is the λ-structure $(p(\Delta^{\pi_u}_{\pi_l}), \sigma', \lambda')$ such that for all $\pi \in \Delta^{\pi_u}_{\pi_l}$:

$$\begin{aligned} \sigma'(p(\pi)) &= \sigma(\pi) \\ \lambda'(p(\pi)) &= \begin{cases} p(\lambda(\pi)) & \text{if } \lambda(\pi) \in \Delta^{\pi_u}_{\pi_l} \\ \text{undefined} & \text{otherwise.} \end{cases} \end{aligned}$$

Let φ be a constraint and $(\mathcal{M}, \alpha)$ a solution of this constraint. The projection α' of α at nodes π_u, π_l is defined iff

$$\alpha(Y) \in \Delta^{\pi_u}_{\pi_l} \cup \{\pi_u\} \text{ for all } Y \in \mathcal{V}(\varphi).$$

In this case, it is defined by

$$\alpha'(Y) = \begin{cases} p(\alpha(Y)) & \text{if } \alpha(Y) \in \Delta^{\pi_u}_{\pi_l} \\ \pi_u & \text{otherwise} \end{cases}$$

for all $Y \in \mathcal{V}(\varphi)$.

Now we can give a formal definition of *intended solutions*. It says that a pair of a λ-structure and a variable assignment is an intended solution of a relaxed constraint iff cutting out the relaxation site yields a solution of the unrelaxed counterpart.

Definition 11 (Intended Solutions). Let φ be a constraint, X a variable and $\mathcal{R}_X(\varphi)$ the relaxation of φ at X. A pair $(\mathcal{M}, \alpha)$ is called an *intended solution* of $\mathcal{R}_X(\varphi)$ iff

1. the projection α' of α at $\alpha(X^u), \alpha(X^l)$ is defined;
2. $(\mathcal{M}^{\alpha(X^u)}_{\alpha(X^l)}, \alpha' \cup \{Y \to p(\alpha(X^l)) \mid Y \in \mathtt{eq}_\varphi(X)\})$ satisfies φ.

Lemma 12. *An intended solution of $\mathcal{R}_X(\varphi)$ is indeed a solution of $\mathcal{R}_X(\varphi)$.*

5.2.2 A Safety Criterion

Given a solution for a relaxed constraint we are now able to decide whether it is intended, i.e. whether it is one where material already present in the constraint has slipped into the gap. We will call a relaxation *safe* if it doesn't have any solution where this is the case.

Definition 13 (Safety). A constraint φ is called *safe at X* iff all solutions of its relaxation $\mathcal{R}_X(\varphi)$ at X are intended.

By this definition, testing whether a certain relaxation is safe would mean showing for each of its solutions that it is intended. Since a constraint may have a lot of solutions, this is not a good option. We would like a criterion that can be stated with reference to only the underspecified semantic representation. Here comes such a criterion, which reduces the expensive testing of all solutions to an entailment problem:

Proposition 14 (Safety Criterion). *Let φ be a constraint, X a variable in φ, and φ' the relaxation of φ at X. Then φ is safe for relaxation at X if the following entailment is true:*

$$\varphi' \models \bigwedge_{Y \in \mathcal{V}(\varphi) \setminus \{X\}} (Y \lhd^* X^u \vee Y \perp X^u \vee X^l \lhd^* Y).$$

Intuitively it is clear why the Safety Criterion should hold: If a node is above, below, or disjoint to the relaxation gap (this is what the disjunction expresses), then, of course, it can't be *in* the gap.

We will say a variable $Y \in \mathcal{V}(\varphi) \setminus \{X\}$ *complies with the Safety Criterion* iff

$$\varphi' \models Y \lhd^* X^u \vee Y \perp X^u \vee X^l \lhd^* Y.$$

Note, that the Safety Criterion does not state an "if-and-only-if" relation. This means that it only filters out a subset of the relaxed constraints which have only intended solutions. It is still strong enough for our purposes, though. We will apply this criterion to semantic representations of sentences containing meaning shifts in the next section. We will also develop some tools for proving its satisfiedness.

5.3 Proving Satisfiedness of the Safety Criterion

In this section, we will apply the Safety Criterion informally to a semantic representation of a sentence involving a case of meaning shift. Then we will develop a tool which supports (formally) proving satisfiedness of the Safety Criterion.

As an example we will use the semantic representation of *Every driver of a mafia boss is parked out back*, which was presented in Figure 5.4 in Section 5.1.1. Does this constraint satisfy the Safety Criterion? Well, it is easy to see that all nodes in fragments II and IV comply with the Safety Criterion; they all either dominate X^u or are disjoint to X^u. The same is true for fragment V: The only node in this fragment is dominated by X^l. To check the nodes of fragments I and III, however, we have to inspect the picture a bit more closely. A hole of fragment II dominates X^u and another hole dominates fragment III. Since this other hole is disjoint to X^u, all nodes of fragment III must be disjoint to X^u as well. That leaves fragment I. The only places this fragment can end up in in a solution is above fragment II or between fragments II and III. In the first case, all of its nodes dominate X^u; in the second case, they are all disjoint to X^u. So the Safety Criterion is satisfied by this constraint.

Now that we have verified that this constraint satisfies the Safety Criterion, can we say anything more general about why this is the case? In fact, we can. This constraint is of a very special structure which has some properties that identify certain places in the constraint as safe for relaxation. This special structure is called a *chain of fragments*, or simply chain. Chains look roughly like depicted in Figure 5.8.

We have a number of upper fragments and a number of lower fragments which

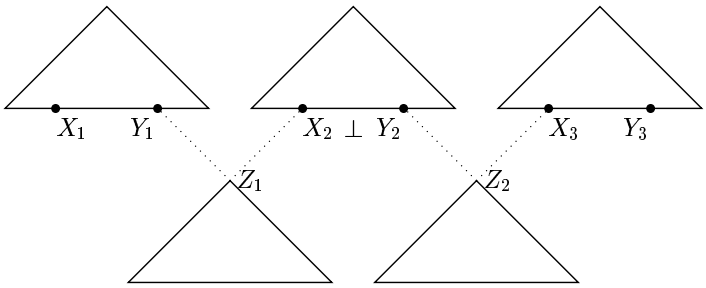


Figure 5.8: A Chain of Fragments

are connected by dominance edges. These dominance edges go from leaves of two different upper fragments to each of the roots of a lower fragment.

More formally:

Definition 15 (Chains). Let φ be a normal constraint, and let $\mathcal{F} = (F_1, ..., F_n)$ and $\mathcal{G} = (G_1, \ldots, G_{n-1})$ be sequences of fragments in φ such that no variable appears in two different fragments. For all i, let X_i, Y_i be different leaves of F_i, and let Z_i be the root of G_i. Then the pair $\mathcal{C} = (\mathcal{F}, \mathcal{G})$ is called a *chain in* φ iff for all $1 \leq i \leq n-1$,

$$Y_i \lhd^* Z_i \wedge X_{i+1} \lhd^* Z_i \text{ in } \varphi.$$

The variables $X_1, Z_1, \ldots, Z_{n-1}, Y_n$ are called the *connection points* of $\mathcal{C}$ and n its *length*.

Chains of fragments have some properties that are very interesting for us, in particular the following two: All connection points of a chain are pairwise disjoint (cf. Proposition 16) and all variables of upper fragments are either disjoint to a connection point or dominate it (cf. Proposition 17).

Proposition 16. *Let $\mathcal{C} = (\mathcal{F}, \mathcal{G})$ be a chain of length n in φ, and let Z_i, Z_k $(0 \leq i < k \leq n)$ be connection points of $\mathcal{C}$. Then*

$$\varphi \models Z_i \perp Z_k \text{ at } \{V_{i+1}, \ldots, V_k\},$$

where the $V_j \in F_j$.

Proposition 17. *Let $\mathcal{C} = (\mathcal{F}, \mathcal{G})$ be a chain in φ, let $F \in \mathcal{F}$ and $G \in \mathcal{G}$, and let Z be the root of G. Then for all $X \in F$,*

$$\varphi \models X \perp Z \vee X \lhd^* Z.$$

These properties are so useful for us, because they give us that, if we relax a chain at a leaf below a connection point all variables of the chain either dominate X^u or are disjoint to X^u. So chains powerfully support the type of inferences we have to draw when checking for satisfiedness of the Safety Criterion.

Returning to the example in Figure 5.4, we can now see that the whole constraint (except for X^l) can be covered by one chain and that the relaxation gap is indeed dominated by a connection point. From this and the fact that X^l dominates itself it follows that the Safety Criterion is satisfied. But one chain is not always enough to cover the entire constraint. Consider, for example, the semantic representation of *Every rich person of a town expects to be on every guest list* depicted in Figure 5.9. A meaning shift takes place to model the fact that what the rich persons expect to be on the guest lists is not themselves but their names.

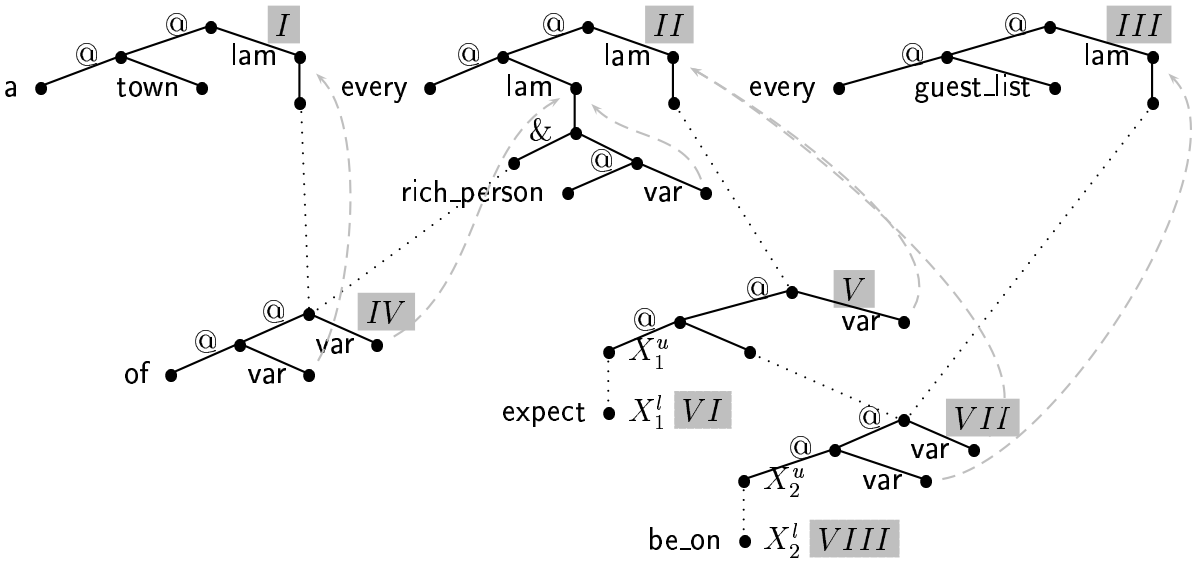


Figure 5.9: *Every rich person of a town expects to be on every guest list*

Here we have an embedded verb phrase because of the control construction. Since every verb introduces a relaxation gap, we have two relaxation gaps in Figure 5.9. We have to show for each of them in turn that the Safety Criterion is satisfied. We will illustrate this for the relaxation gap in fragment V. We cannot find a chain which has a connection point that dominates the relaxation gap and which covers the entire constraint in one go. But we can find several chains which all have a connection point that dominates the relaxation site and which, taken together, do cover the entire constraint. More concretely, we need two chains, namely the one with fragments I and II as upper fragments and IV as lower fragment and the one which consists of fragments III and V as upper and VI as lower fragments. Because of Propositions 16 and 17, these two chains let us conclude that all nodes of the constraint graph (except for X^l) either dominate X^u or are disjoint to X^u, so that the Safety Criterion is satisfied for the relaxation gap in fragment V. Proving satisfiedness of the Safety Criterion for the relaxation gap in fragment VI works in just the same way.

5.4 A Safe Syntax/Semantics Interface

The same technique of intelligently covering the entire constraint with chains can now be used to show that it is not a special property of the constraints in Figure 5.4 and 5.9 that they are safe, but that in fact our syntax/semantics interface derives only safe constraints. This means that we can relax our semantic representation to model meaning shift at exactly those positions that we chose for linguistic reasons. We don't have to fear any kind of overgeneration due to this relaxation.

Note that any constraint derived by the relaxed syntax/semantics interface, i.e. rules (b1) – (b9) and rules (b10') and (b10"), can also be constructed from a constraint derived by the unrelaxed syntax/semantics interface, i.e. rules (b1) – (b10), by a sequence of relaxation operations. The main result we will prove is the following:

Theorem 18. *Let φ be a constraint which was constructed by the unrelaxed syntax/semantics interface of the example grammar and let φ' be a constraint which was constructed for the same utterance by the relaxed syntax/semantics interface. Let X be a node in φ such that there is a relaxation gap at the*

corresponding position in φ'. Then the Safety Criterion is satisfied for the relaxation of φ at X.

Looking at the general structure of the constraints derived by our syntax/semantics interface which is as shown in Figure 5.10 will intuitively make clear that this theorem holds. The picture suggests that for each relaxation site in the displayed constraint we can cover the constraint with chains which have a connection point that dominates the relaxation site.

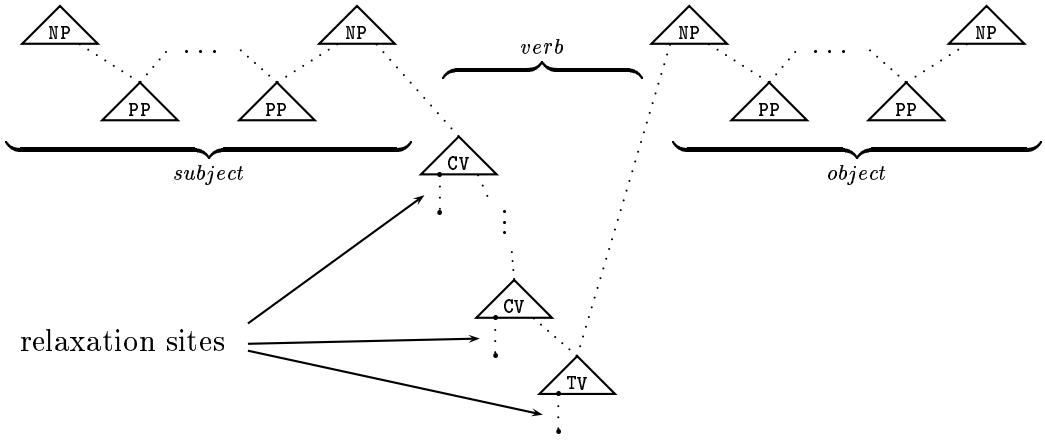


Figure 5.10: Output of the Syntax/Semantics Interface: Schematic View

For proving Theorem 18, we will proceed in three steps. We will first show that the semantic contribution of noun phrases are chains. Then we will make a similar statement about the contribution of verb phrases. Finally, we will use these results to prove the theorem.

Lemma 19. *Let T be a parse tree of our grammar, and let φ be the constraint that our syntax/semantics interface assigns to it. Furthermore, let t be a subtree of T whose root is labeled with* NP*, let n be the number of* NP *nodes in t, and let φ_t be the conjunction of the constraints corresponding to the nodes of t. Then there is a chain $\mathcal{C} = ((F_1, \ldots, F_n), (G_1, \ldots, \mathcal{G}_{n-1}))$ of length n covering φ:*

$$\mathcal{V}(\varphi_t) = \bigcup_{F \in \mathcal{F}} F \cup \bigcup_{G \in \mathcal{G}} G.$$

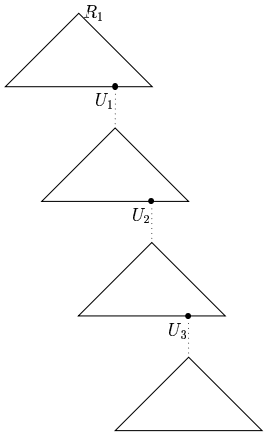


Figure 5.11: A Cascade of Fragments

It would be nice if we could say something similar about the general structure of the semantic contribution of verb phrases. In fact we can, but we first need the notion of a *cascade of fragments*.

Definition 20 (Cascade of Fragments). Let φ be a normal constraint and let $\mathcal{CF} = (S_1, \ldots, S_n)$ be a sequence of fragments in φ such that no variable appears in two different fragments. For all i let R_i be the root and U_i a leaf of S_i. $\mathcal{CF}$ is called a *cascade of fragments in* φ iff for all $1 \leq i \leq n-1$,

$$U_i \lhd^* R_{i+1} \text{ in } \varphi.$$

We will call $R_1, U_1, \ldots, U_{n-1}$ the anchoring points of $\mathcal{CF}$. n is the length of the cascade. Figure 5.11 shows a cascade of length 4.

The following property of cascades is important for us.

Proposition 21. *Let $\mathcal{CF} = (S_1, \ldots, S_n)$ be a cascade of fragments in φ, let $R_1, \ldots, R_n$ be the roots of $S_1, \ldots, S_n$, and let $U_0, \ldots, U_{n-1}$ be the anchoring points of $\mathcal{CF}$. Then*

$$\varphi \models U_i \lhd^* X$$

for any $X \in S_{i+1} \cup \ldots \cup S_n$ and

$$\varphi \models X \lhd^* U_i \vee X \perp U_i$$

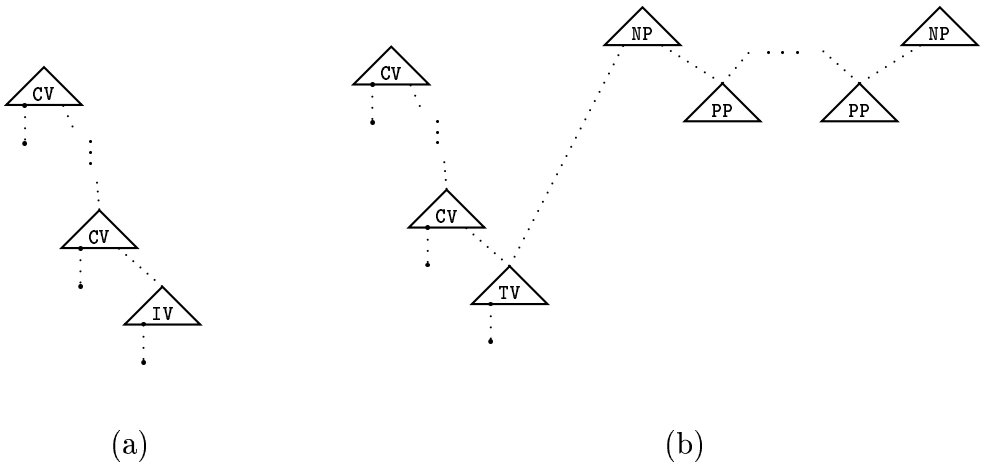


Figure 5.12: Semantic Contribution of Verb Phrases

for any $X \in S_1 \cup \ldots \mathcal{S}_i$.

Now, we can describe the semantic contribution of verb phrases in a general way. There are two possibilities of what it may look like. Figure 5.12 shows that they are essentially put together from cascades and chains of fragments. There is one additional node "dangling" from each fragment of the cascade which is the lower variable of a relaxation gap. The leaf such an extra variable is dominated by is disjoint to the anchoring point of the fragment. Note that with respect to the properties of cascades stated in Proposition 21 these variables can be treated as a variable of the fragment they are "dangling" from.

Lemma 22. *Let T be a parse tree of our grammar and let φ be the constraint that our syntax/semantics interface assigns to it. Furthermore, let t be a subtree of T whose root node ν is labeled with* `VP`*, let n be the number of verb nodes, i.e.* `IV`, `TV`, *and* `CV` *nodes, in t, and let φ_t be the conjunction of the constraints corresponding to the nodes of t. Then there are two possibilities for what the structure of φ_t may look like.*

1. *There is a cascade of fragments $(S_1, \ldots, S_n)$ with anchoring points*

$U_0, \ldots, U_n$, *where* $n \geq 1$ *is the number of verb nodes in* t. X_ν *(which is the variable in the semantic representation associated with the node* ν *of the parse tree, i.e. the root of* t*) is the root of fragment* S_1. *This is depicted in Figure 5.12 (a).*

2. *There is a cascade of fragments* $(S_1, \ldots, S_n)$ *with anchoring points* $U_0, \ldots, U_n$, *where* $n \geq 1$ *is the number of verb nodes in* t. X_ν *is the root of fragment* S_1. *Furthermore, there is a chain of fragments* $((F_1, \ldots, F_m), (G_1, \ldots, \mathcal{G}_{m-1}))$, $m \geq 1$, *and the root of fragment* S_n *is dominated by a hole of fragment* F_1. *This is depicted in Figure 5.12 (b).*

No variable of φ_t *takes part in more than one of these fragments and all variables except for* $X_1^l, \ldots, X_n^l$ *take part in at least one of the fragments.* X_i^l *is dominated by a leaf of fragment* S_i *which is disjoint to* U_i.

Now, everything is prepared for proving Theorem 18. We will show that every variable of φ' complies with the Safety Criterion with respect to any relaxation site in φ'. The fact that all constraints derived by our syntax/semantics interfaces a are constructed from chains and cascades of fragments, allows us to exploit Propositions 16, 17 and 21, which make powerful statements about the relative position of different variables within one chain or cascade respectively.

Proof of Theorem 18. The only way our grammar allows to expand `S`, is to `NP VP`. So any parse tree T has the following structure: $T = (\texttt{S}(\texttt{NP}\ t')(\texttt{VP}\ t''))$. Let φ be the constraint that the relaxed syntax/semantics interface derives for T. By Lemma 19 the semantic contribution of subtree t' is a chain and the semantic contribution of t'' is as one of the two cases described by Lemma 22.

case 1:. The semantic contribution of t'' is as in case 1.) of Lemma 22, i.e. we have a chain $\mathcal{C} = ((F_1, F_2, \ldots, F_n), (G_1, \ldots, G_{n-1}))$ and a cascade $\mathcal{CF} = (S_1, \ldots, S_m)$ with anchoring points $(U_0, \ldots, U_{m-1})$. The proof will be achieved in two steps. First we will show that all variable of the chain comply with the Safety Criterion wrt. any relaxation site in the cascade and then we will show this for all variables of the cascade.

Rule (b1) of the syntax/semantics interface connects the chain and the cascade in such a way that the result is the chain $\mathcal{C}' = ((F_1', F_2, \ldots, F_n),$

$(G_1, \ldots, G_{n-1})$) with connection points $Z_0, \ldots, Z_n$. $Z_0 \in F_1'$ dominates the root of S_1. By Propositions 16 and 17 we know that any variable of the chain dominates Z_0 or is disjoint to it. We can furthermore conclude from Proposition 21 and the fact that Z_0 dominates the root of S_1 that Z_0 also dominates all variables of the cascade. So for any variable X of the chain and any variable Y of the cascade

$$\varphi \models X \lhd^* Y \ \vee \ X \perp Y$$

holds. This gives us that all variables of the chain comply with the Safety Criterion with respect to any relaxation site, since relaxation sites occur only in the cascade. Now, we have to show that all variables of the cascade also comply with the Safety Criterion wrt. any relaxation site. Let X_i^u be the upper variable of the relaxation site in fragment S_i. X_i^u is disjoint to U_i. By Proposition 21 U_i dominates all variables of fragments $S_{i+1}, \ldots, S_m$. Therefore, X_i^u is disjoint to all variables of fragments $S_{i+1}, \ldots, S_m$. Since X_i^u is a leaf of fragment S_i all variables of S_i dominate X_i^u or are disjoint to it. Furthermore, also by Proposition 21, U_{i-1} dominates X_i^u and all variables of fragments $S_1, \ldots, S_{i-1}$ dominate U_{i-1} or are disjoint to it. This means that the same holds for all variables of fragments $S_1, \ldots, S_{i-1}$ with respect to X_i^u. Hence

$$\varphi \models X \lhd^* X_i^u \ \vee \ X \perp X_i^u$$

for $X \in S-1 \cup \ldots \cup S_m$ and any $i, 1 \leq i \leq m$.

case 2:. The semantic contribution of t'' is as in case 2.) of Lemma 22. We will need three steps to complete the proof for this case. The first step, where it is shown for all variables of the semantic contribution of the subject that they comply with the Safety Criterion, is as in case 1.). The second step, where all variables belonging to the cascade are shown to comply with the Safety Criterion, is also as above. That leaves the chain introduced by the object noun phrase. Let $\mathcal{CF} = (S_1, \ldots, S_m)$ be the cascade. Its anchoring points are $(U_0, \ldots, U_{m-1})$ and $X_1^u, \ldots, X_m^u$ are the upper variables of the relaxation gaps in fragments $S_1, \ldots, S_m$. Furthermore, let $\mathcal{C} = ((F_1, \ldots, F_n)(G_1, \ldots, G_{n-1})$ be the chain and Z_0 the leaf of fragment F_1 which dominates the root of S_m. Note that Z_0 is a connection point of $\mathcal{C}$. So X_m^u is dominated by a connection point and the claim holds with respect to this relaxation site. For any

other relaxation site X_i^u the chain can be expanded in the following way: $\mathcal{C}' = ((S_i, F_1, \ldots, F_n)(S_m, G_1, \ldots, G_{n-1}))$. This is a chain because there is a leaf of fragment S_i, namely U_i, which dominates the root of S_m (Proposition 21). X_i^u is disjoint to U_i and therefore a connection point of $\mathcal{C}'$. Hence, by Propositions 16 and 17, all variables of $\mathcal{C}$ comply with the Safety Criterion with respect to this relaxation site.

□

This theorem gives us for that no relaxation site in a constraint derived by the relaxed version of our syntax/semantics interface will ever have any unintended solutions. This means that relaxation for modeling meaning shifts is safe: we can freely relax whenever we need space for the insertion of a shift operator.

5.5 Summary

In this chapter, we presented a treatment of meaning shifts in lexical semantics which was first proposed by Egg. By using dominance constraints as a meta-language he describes phenomena of meaning shift by relaxed underspecified representations which leave open whether a shift operator is introduced and what it exactly looks like. These decisions which have to be based on pragmatic as well as lexical knowledge can thus be postponed as long as it is necessary. Whenever the missing information can be determined it can be added monotonically.

We have adapted the syntax/semantics interface of the previous chapter to derive relaxed underspecified representations in line with Egg´s approach.

Egg´s approach relies mainly on making underspecified representations even less specific. As we have seen, the danger of overgeneration is inherent to such an approach. However, we could prove that all constraints derived by our syntax/semantics interface are safe in so far as their structure prevents any overgeneration. This proof relied heavily on the concept of chains of fragments. A chain is a substructure of a constraint graph with some special properties that nicely supports the type of inferences we had to make.

Chapter 6

Lots of Questions and Some Answers

In the last chapter we developed a framework for modeling meaning shifts which provides a semantic representation that can monotonically be augmented with new material at any stage of processing an utterance. We argued in the previous chapter that such a framework is exactly what we need. Given this framework the obvious next step is to describe how the insertion of new material into the semantic representation can be controlled. This brings us back to the questions we raised in Chapter 2, namely (1) when to add a shift operator, (2) where to add it, and (3) what shift operator to add. We saw in Chapter 3 that various people have attempted to answer this question (among them Dölling (1994, Copestake and Briscoe (1995, Copestake and Lascarides (1998, Hobbs et al. (1993, Nunberg (1995)). (Egg 1999) himself largely ignores the control aspect and just assumes some pragmatic component which comes up with the appropriate shift operators. In Chapter 2 we saw some evidence, however, that what is an appropriate shift operators cannot be explained in purely pragmatic terms. We will now inspect these questions with respect to Egg's accounts and suggest how the influence of lexical and sortal information on meaning shift could be accounted for.

6.1 When to Add a Shift Operator

In almost all examples we saw so far meaning shift was a way of avoiding a sort or type conflict. Consider for example the semantic representation for the by now well-known sentence *I am parked out back* displayed in Figure 6.1. The nodes of this graph are labeled with sorted types which is to be interpreted in the following way: Type α at node X says that the expression which is described by the (sub)tree under node X has type α. If the dominance edge at the position marked with $\not\downarrow$ would just be replaced by identity of the nodes X^u and X^l a sort clash would occur, because *human* and *vehicle* are incompatible.

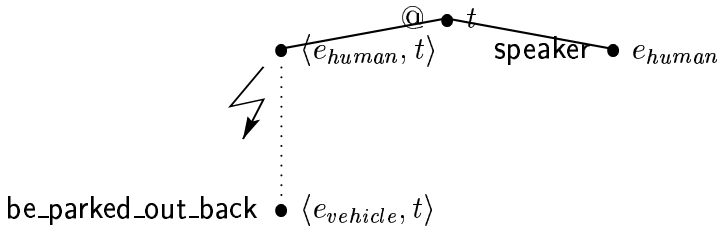
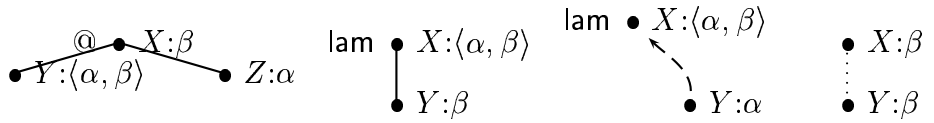


Figure 6.1: Semantic Representation With Sorted Types

The types of a node can be determined by simple type inference. Semantic constants are of the usual types, as indicated by the following examples.

a	$\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$
man	$\langle e, t\rangle$
$love$	$\langle e, \langle e, t\rangle\rangle$

Furthermore, there are constraints on how the types in a local subtree of a semantic representation are related to each other. For semantic representations we have seen so far it suffices to consider the four local structures displayed in Figure 6.2. Figure 6.2(a), for example, says that if something of type $\langle\alpha, \beta\rangle$ is applied to something of type α, the result is of type β. Note that there are no constraints on how the types of nodes X and Y in Figure 6.2(d) are related. They are only connected by a dominance edge, which means that any material changing the type of Y could intervene. This is, in fact, what we exploited when relaxing semantic representations to avoid type or sort conflicts.



(a) (b) (c) (d)

Figure 6.2: Local Constraints on Types

The type information available from the semantic constants can be percolated through the tree (by unification) according to the rules stated in Figure 6.2. This also holds if we enrich our type system with sorts (as we already did in Figure 6.1). Actually, for now, we only want to assign sorts to things of type e. These sorts are partially ordered on an ontological hierarchy. The lattice in Figure 6.3 might be an excerpt of such a hierarchy.

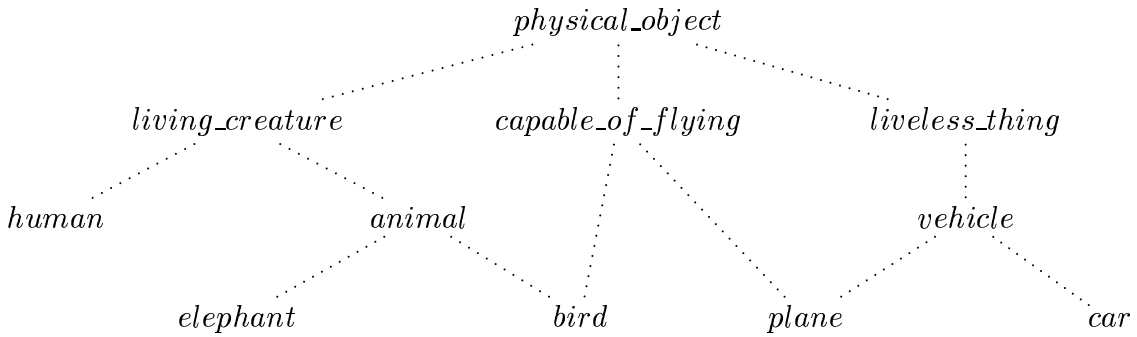


Figure 6.3: Excerpt of an Ontology

Unification of sorted types is done in the usual way: Namely, the result of unifying types e_i and e_j is e_k where k is the greatest lower bound of i and j; if there is no such k unification fails (see e.g. (Copestake et al. 1993)).

The following examples (Figures 6.4, 6.5, and 6.6) will illustrate this mechanism. The lexicon provides sorted types for *a*, *bird*, and *fly*, namely $\langle\langle e_s, t\rangle, \langle\langle e_s, t\rangle, t\rangle\rangle$ for some sort s, $\langle e_{bird}, t\rangle$, and $\langle e_{capable\_of\_flying}, t\rangle$. This sort and type information is then percolated through the tree by the rules given

in Figure 6.2. At some point during the construction of the semantic representation for *A bird flies* $a(bird)$ of type $\langle\langle e_{bird}, t\rangle, t\rangle$ and $\lambda x.fly(x)$ of type $\langle e_{capable\_of\_flying}, t\rangle$ are combined. In order to do so, e_{bird} and $e_{capable\_of\_flying}$ have to be unified. This results in e_{bird}, since *bird* is the greatest lower bound of *bird* and *capable_of_flying*, and the semantic representation presented in Figure 6.4.

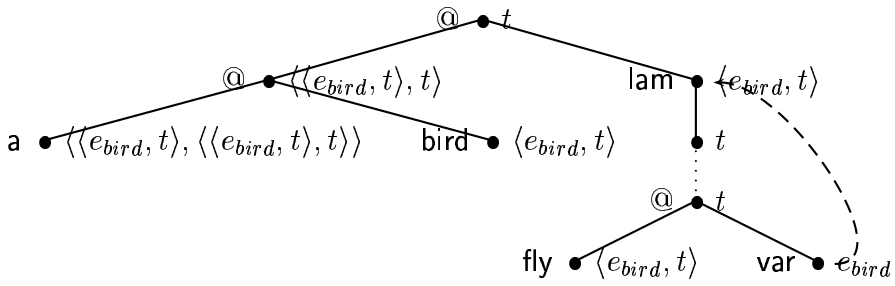


Figure 6.4: Example 1: *A bird flies*

The result of semantic construction for the sentence *An animal flies* is shown in Figure 6.5. The unification of e_{animal} with $e_{capable\_of\_flying}$ constraints the animal which is flying to be a bird.

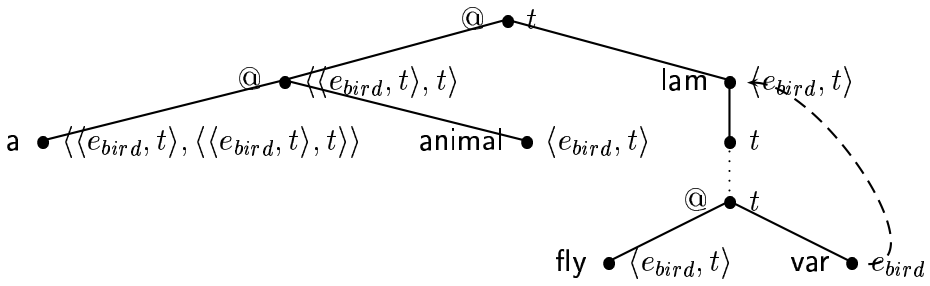


Figure 6.5: Example 2: *An animal flies*

In contrast, *An elephant flies* gives rise to a sort conflict (again indicated by ↯), because $e_{elephant}$ and $e_{capable\_of\_flying}$ are incompatible (cf. Figure 6.6).

In addition to the cases where a meaning shift is necessary to resolve a sort or type conflict, there are cases where even without adding a shift operator the

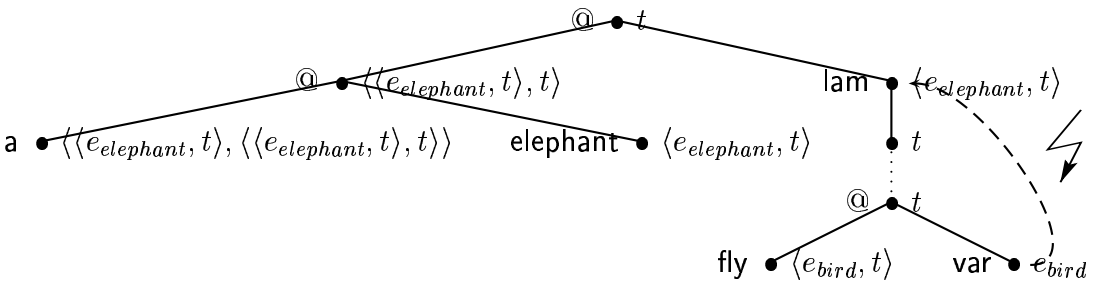


Figure 6.6: Example 3: *An elephant flies*

semantic representation was completely well-typed and sorted, but still the way they are usually understood involves a meaning shift. We saw one such example in Section 2.2.1 of Chapter 2. These appear to be rarer though and are sometimes misunderstood which leads to funny incidents[1]. In cases like this insertion of a shift operator is triggered by context or world knowledge. How this exactly work, e.g. what kind of inferences are made, is still an open question.

6.2 Where to Add the Shift Operator

We explicitly introduced the places where inserting a shift operator is possible by relaxing the semantic representations at these positions. The syntax/semantics interface we presented in the previous chapter provides relaxation gaps above all predicates introduced by verbs. If meaning shift is necessary to prevent a sort or type conflict those relaxation gaps where a shift operator has to be introduced can be distinguished by the fact that the type of the upper variable of the relaxation gap and the type of the lower variable are not compatible. This is for example the case in the semantic representation of *I am parked out back* in Figure 6.1. In cases of pragmatically

[1]Like the one about the person who called a Nintendo hot line for help with a difficult situation in computer game and was told to kneel down and pray at a certain point of the game. He didn't carry out the meaning shift and complained, because it didn't work.

triggered meaning shift the knowledge which tells us that a certain reading is improbable can be used to infer where the incompatibility is, i.e. to which predicate a shift operator should be applied.

Note, however, that our syntax/semantics interface restricts meaning shift to verbs, while we argued in Chapter 2 that there are also cases where not the meaning of a verb but the meaning of a noun is shifted. *The ham sandwich at table 6 is getting impatient* is an example of this type of meaning shift. To account for these cases as well we would have to further relax the semantic representations. More specifically, we would have to introduce relaxation gaps above nodes labeled by predicates corresponding to the meaning of a nouns. The semantic representation of *A mafia boss is parked out back* would then look like depicted in Figure 6.7.

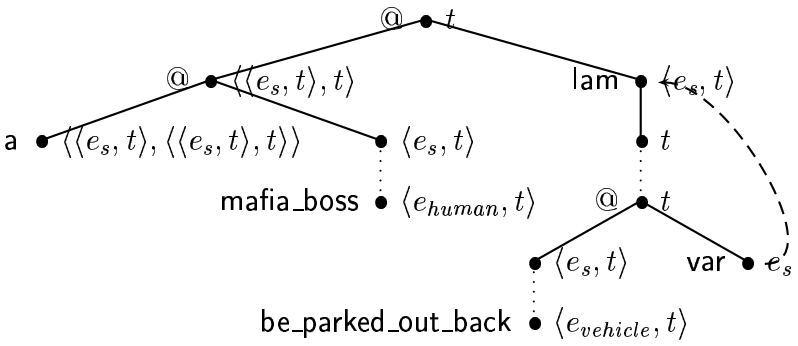


Figure 6.7: The Effect of Further Relaxation

The inferences in sort and type information which this representation allows us to draw are not strong enough anymore to unambiguously locate the position where the shift operator has to be introduced. Nevertheless, the application of a shift operator to *mafia_boss* is ruled out in this example by the fact that there is no appropriate operator for a meaning shift of this type, i.e. an operator shifting a human being to a vehicle. In fact, as we said in Chapter 2, meaning shifts to the verb seem to be the standard case, while meaning shifts of the noun are restricted to specific situations, in which these meaning shifts have a high degree of conventionality. This suggest that in many situations a meaning shift of the noun is simply not possible, because no appropriate operator is available, and in those situations where such an operator is available, it should probably be preferred to others. These

observations are still quite vague and need closer inspection.

The question of how to control the search for an appropriate operator seems to become even more problematic when cases where both, a meaning shift of a noun as well as a verb, take place, as it is the case in the following example:

(6.1) The ham sandwich is parked out back.

6.3 What Shift Operator to Add

This section is concerned with the question of what information plays a role in meaning shift and how an appropriate shift operator can be determined. As argued in Chapter 2 this might involve knowledge which does not stem from linguistic sources. It is not clear how the shift operator is determined when several sources of knowledge have to interact. As we saw in Chapter 3 it has been proposed to use abductive inference (Hobbs et al. 1993) or a form of default reasoning (Copestake and Lascarides 1998). In order to yield a full account of meaning shift something along these lines has to be provided for our framework of underspecified semantic representations as well. In particular, inference mechanisms that work on underspecified descriptions of logic formulas are desirable. The examples we saw suggest that scope ambiguities (and maybe other ambiguities as well) are independent of meaning shift, so that even in not yet fully specified semantic representations there is enough information to decide on an appropriate shift operator. Such mechanisms of direct deduction would allow to carry out the process of meaning shift on underspecified representations, which is clearly in the spirit of using information whenever it is available and only enumerating readings when it is absolutely necessary. Unfortunately, direct deduction mechanisms are still in their infancy, but recent results (Jaspers and Koller 1999) let hope for more soon. Here, I will not go into the question of how context and pragmatic knowledge influence meaning shift, though. I will concentrate on the role of more tractable information such as lexical and sortal kmowledge.

We saw in Chapter 2, that there are classes of meaning shift phenomena, namely polysemy and certain types of metonymy, which exhibit a high degree of conventionality. This was suggested to be due to the fact that in these cases the shift operator is lexically determined and thus no pragmatic or world knowledge is necessary for finding it.

Let's first have a look at polysemy. We saw in Chapter 3 that both Dölling (1994) and Pustejovsky (1995) a special kind of sorts to account for the polymorphic behavior of polysemic words. Dölling calls these sorts *hybrid sorts* while Pustejovsky uses the term *dotted sorts*. In both approaches these special sorts are constructed by combining other sorts, in particularly sorts which correspond to classes of the ontology which are usually not compatible. The word *door* is taken to denote objects of a dotted sort (or hybrid sort) which was constructed by combining the sort *aperture* with the sort *physical object*. In the real world these two sorts are incompatible, since no object can be both, an aperture and a physical object. Looking more closely at what classifies objects as doors, we find that, in fact, doors are not *one* thing which is both, an aperture and a physical object at the same time, but rather consist of *two* components where one is an aperture and the other one a physical object. These two components are both so important for the definition of *door* that normally we refer to both when saying *door*, so that it is plausible to assume some kind of combination of *aperture* and *physical object* as the sort of *door*-objects. So the linguistic sort hierarchy is richer than an ontology which is reflecting the classes of object that exist in the real world, since it contains dotted sorts[2] which don't correspond to a class in the ontology.

A question which immediately arises is how dotted sort relate to their components. All examples of polysemy we have seen seem to imply that an object with a dotted sort has all properties of either one of its component. For convenience some of the examples are repeated in 6.2 below.

(6.2) (a) Mary walked through the door.

(b) Mary painted the door blue.

(c) A dwarf stepped out of the blue door.

This suggests that dotted sorts could be taken to be subsorts of their components. With this approach the sentences in 6.2 do not give rise to any sort conflict. By the mechanisms presented in Section 6.1 of this chapter the predicates are simply specified to denote a property of objects of sort *aperture* $\cdot$ *physical_object* instead of sort *aperture* or *physical_object* respec-

[2]I will adopt Pustejovky's term and also his notation, writing $s_1 \cdot s_2$ for the dotted sort consisting of a combination of sorts s_1 and s_2.

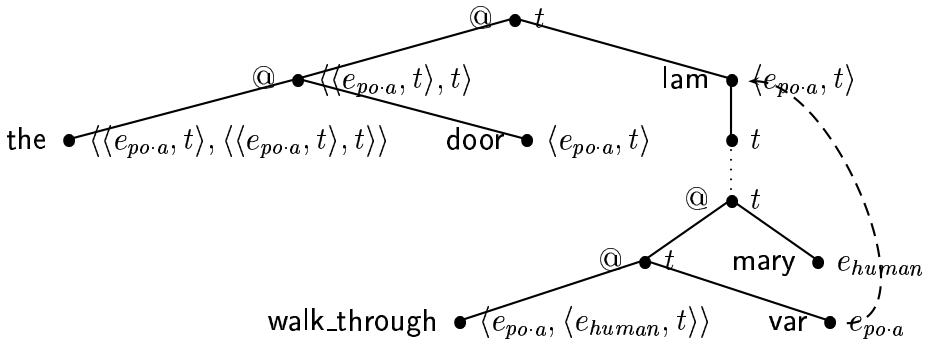


Figure 6.8: Dotted Sorts as Subtypes of Their Components

tively. The semantic representation of *Mary walked through the door* looks as shown by Figure 6.8, where $e_{po \cdot a}$ stands for $e_{physical\_object \cdot aperture}$.

What is problematic about this approach is that it implies that dotted sorts are more specific than their components. This is counterintuitive. If there are no contextual restrictions, polysemic words denote dotted sorts, and only if the situation imposes constraints on the interpretation, one component may be singled out. In a conversation about book covers, for instance, an utterance like *This book is interesting* is not vague, but unambiguously making a statement about the physical object aspect of book. On the other hand, dotted sorts cannot be taken to be more general than their components, since this would mean that every aperture was also a door, which is of course not true. This suggests that dotted sorts cannot easily be incorporated into the hierarchy of simple sorts. So let's assume that they are ordered in a hierarchy of their own. In this hierarchy, $i \cdot j$ subsumes $i' \cdot j'$ iff i subsumes i' and j subsumes j' in the hierarchy of simple sorts. Let's also define the function COMPONENTS in such a way that when given a dotted sort as argument, it returns the set of all simple sorts constituting this dotted sort and the dotted sort itself; when given a simple sort, it returns the set containing only the simple sort itself.

$$COMPONENTS(i) = \{j \mid j = i \text{ or } j = j', \text{ where } i \text{ is a dotted sort and } \exists k \text{ such that } i = k \cdot j' \text{ or } i = j' \cdot k$$

This approach leads to sort conflicts, when construction semantic representations for sentences as those given in 6.2. Let's have a look at an example. The (unrelaxed) semantic representation including sorts for *Mary walked through the door* looks as presented in Figure 6.9.

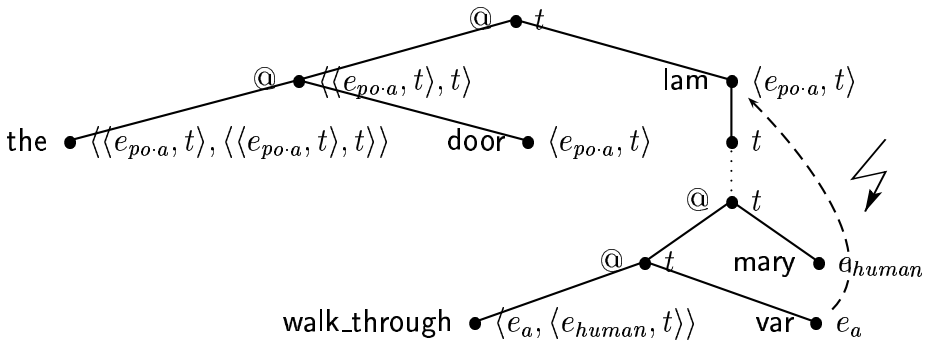


Figure 6.9: A Sort Conflict Due to a Dotted Sort

There is a sort conflict, since *paint_blue* asks for an argument of sort *physical_object*, what is supplied is a dot object of sort *physical_object·aperture*, and neither one subsumes the other. Hence, we have to fill in an operator somewhere to resolve this conflict and to account for the meaning shift which takes place.

To keep the process of meaning shift as uniform as possible, we would like to have a shift operator which takes *walks_through* as input and yields something asking for an argument with a dotted sort. Intuitively, the final semantic representation should look like in Figure 6.10, where P is some operator.

P shifted the denotation of *walk_through* from a set M of tuples of objects of sorts *aperture* and *human* to a set M' of tuples of objects of sorts *physical_objects* $\cdot$ *aperture* and *human*. Any element $\langle u', v\rangle$ of M' is such that there is an object u of sort *aperture* and $\langle u, v\rangle \in M$ and u is the aperture of the door u'. A general shift operator of this kind can be characterized as follows:

$$P: \quad << e_i, < e_j, t >>, < e_{i'}, < e_{j'}, t >>>, \text{ where}$$
$$i \in COMPONENTS(i') \text{ and } j \in COMPONENTS(j')$$

This operator applies only to binary predicates, but of course similar operators can be defined for predicates with another arity.

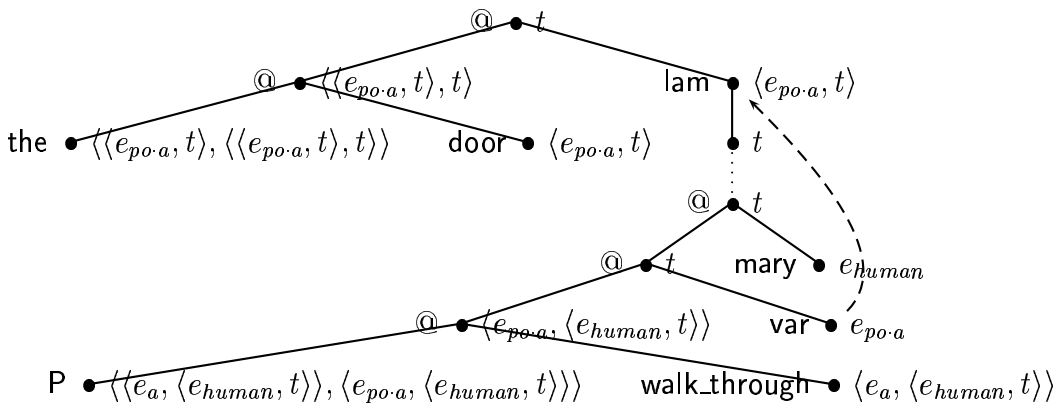


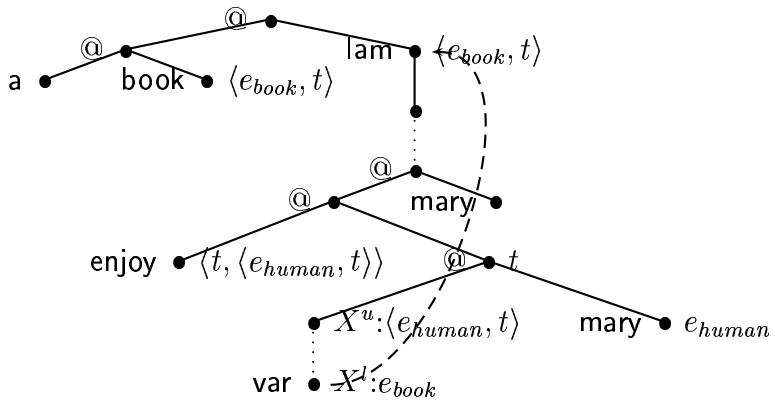
Figure 6.10: Resolving a Sort Conflict Due to a Dotted Sort

Now, let's turn to the type of meaning shift which is due to metonymy. We saw in Chapter 2 that there is some evidence for lexical information to be a factor in determining the shift operator. We argued there that the default interpretation of *Mary enjoyed a book* is *Mary enjoyed reading a book* and that the information that what is usually done to books is reading is lexical information. Pustejovsky (1995) accounts for this by means of his qualia structure (cf. Chapter 3, Section 3.3). The qualia structure allows to specify certain kinds of information related to the meaning of a word within its lexical entry. All these qualia roles are available as a semantic representation of the corresponding word. Pustejovsky assumes that one of them is selected by an operator introduced into the semantic representation, but what triggers this process and determines which qualia roles is selected is not described in detail. (Copestake and Briscoe 1995) also adopt his idea of a qualia structure to specify additional semantic information in the lexical entry. However, in their account, the appropriate qualia role is selected already during the semantic construction process (cf. Chapter 3, Section 3.4). This can result in decisions which later turn out to be wrong and have to be revised. We will now sketch how the notion of qualia structure could be adapted to Egg's approach.

We will think of qualia information as being associated with objects of a certain sort. For instance, if we have an object of sort *book*, the information that it can be read is provided by the qualia structure associated with the sort

book. This information can be incorporated into the sort hierarchies. This then calls for some kind of default inheritance, in order to account for the fact that *dictionary* should be a subsort of *book*, although they are usually rather referred to than read.

So, considering an example, the relaxed semantic representation for *Mary enjoyed the book* would look as depicted in Figure 6.11. Note that the syntax/semantics interface given in the previous chapter does not handle this case, but it could easily be extended in order to do so. The sort associated with X^l, the lower variable of the relaxation site, gives us access to the qualia information of that sort. This qualia information provides the preferred shift operators. If there is no contradictory contextual or pragmatic information, the shift operator is chosen from this set.



$$qualia(book) = \begin{bmatrix} \text{telic} & read{:}\langle e_{book}, \langle e_{human}, t\rangle\rangle \\ \text{agentive} & write{:}\langle e_{book}, \langle e_{human}, t\rangle\rangle \end{bmatrix}$$

Figure 6.11: *Mary enjoyed the book*

6.4 Summary

This chapter made clear that the question what controls meaning shift and how this can be modeled is still largely unsolved. However, partial answeres exploiting in particular lexical and sortal information, have been given. We

sketched how the influence of lexical and sortal information on meaning shift can be accounted for in the underspecified approach we presented in the previous chapters. To this end we made use of dotted sorts and qualia information.

Chapter 7

Conclusion

In this thesis, we have examined an underspecified treatment of *meaning shift* due to phenomena like polysemy or metonymy. The key idea of such a treatment is to cushion potential type or sort conflicts by *relaxing* the semantic representation of a sentence involving such a meaning shift so that it can be monotonically augmented with an operator which makes the meaning shift explicit.

The framework CLLS (Constraint Language for Lambda-Structures), which we used as semantic representation language, exploits dominance constraints between nodes to arrive at underspecified tree descriptions. We presented a syntax/semantics interface which derives CLLS constraints as underspecified semantic representations that allow for the augmentation with shift operators. We showed that this syntax/semantics interface is *safe*, i.e. none of the constraints derived by it have solutions where material of the original constraint appears at the position of the relaxation gap. In proving this result we developed the notion of *chains of fragments*. This is a novel subclass of dominance constraints which has a number of interesting properties. In addition to proving safety of a constraint they can be used show that the syntax/semantics interface produces only solvable constraints, and a variation of chains can be used to show that they are efficiently solvable. We consid-

ered only the syntax/semantics interface for a grammar fragment, but we are confident that the results should scale up to a grammar with a much wider coverage (involving e.g. relative clauses, embedded sentences, etc.). What is much more unclear and should be a point of investigation in the future is, if the results can be adapted to the full framework of CLLS which provides parallelism constraints.

As already pointed out several times, this is only the first step in a comprehensive treatment of meaning shift. The discussion in Chapter 6 indicated what the next step involving the actual insertion of an appropriate shift operator could look like, but also showed that there are still many open questions related to controlling the search space for the shift operator. In particular, the role pragmatics plays and how the necessary inferences can be made on underspecified semantic representations remains a problem for future research.

Appendix A

Proofs

A.1 Properties of Fragments

This section will supply the missing proofs for Lemmas 5 – 7 of Chapter 4.

Lemma 5. Let φ be a normal constraint, let F be a fragment in φ, let X, Y be different variables in F, and let $(\mathcal{M}, \alpha)$ be a solution of φ. Then there is a $W \in F$ such that the following entailment holds:

$$\varphi \models X \lhd^* Y \vee Y \lhd^* X \vee X \perp Y \text{ at } \{W\}.$$

Proof. Assume that $(\mathcal{M}, \alpha)$ satisfies φ; we show that it must also satisfy the right-hand side. The proof is by induction over the number n of labeling constraints that ensure the connectedness of the fragment. As φ is normal, there is at most one labeling constraint that has a given variable as its head, and the set of connecting labeling constraints is always uniquely determined.

$\underline{n = 1.}$ Since X and Y must be connected, φ either contains $X{:}f(\ldots Y \ldots)$ or $Y{:}f(\ldots X \ldots)$.

$\underline{n \to n+1.}$ Let F', F be fragments in φ which are connected via n and $n+1$ labeling constraints, respectively. Furthermore, let $F' \subseteq F$. Then there must be a variable $Y' \in F'$ and a labeling constraint of the form $Y'{:}f(Y_1, \ldots, Y, \ldots, Y_k)$ or $Y{:}f(Y_1, \ldots, Y', \ldots, Y_k)$ in φ, where $Y \in F$ but $Y \notin F'$, which is needed to connect F but not F'. Note that $F = F' \cup \{Y_1, \ldots, Y_k\}$.

We first consider the case that $Y'{:}f(\ldots Y \ldots)$ is in φ. By the induction hypothesis, there are three possibilities of how the denotation of a variable $X \in F'$ can be related to that of Y'.

1. If $X \lhd^* Y'$, then X dominates Y and all of the Y_i.
2. $Y' \lhd^* X$ is not possible, since then Y' would have two labels.
3. If $X \perp Y'$ at $\{W'\}$ then $X \perp Y$ at $\{W'\}$, and for all i, $X \perp Y_i$ at $\{W'\}$ as well.

Now consider the case that $Y{:}f(\ldots Y' \ldots)$ is in φ. We make the same case distinction as above.

1. If $X \lhd^* Y'$, then X and Y' must denote the same node (a case which is covered by the next case), for assume otherwise. Then because of the connectedness of F', there is a sequence of labeling constraints that connects X and Y'. Hence, there is an $Y'' \in F'$ such that $Y''{:}g(\ldots, Y', \ldots)$ in φ, in contradiction to normality and satisfiedness of φ.
2. If $Y' \lhd^* X$, then $Y \lhd^* X$, and for all i, $Y_i \perp X$ at $\{Y\}$.
3. $X \perp Y'$ at $\{W'\}$ is impossible. This requires that there is a labeling constraint $W'{:}g(\ldots, W_1, \ldots, W_2, \ldots)$ such that $W_1 \lhd^* Y'$ and $W_2 \lhd^* X$; but this implies that there is a sequence of labeling constraints that connects W_1 and Y', which leads to a similar contradiction as in the first case.

□

Lemma 6. Let φ be a normal constraint, let F be a fragment in φ, and let $(\mathcal{M}, \alpha)$ be a solution of φ. Then F contains exactly one variable R such that

$$\varphi \models \bigwedge_{X \in F} R \lhd^* X.$$

Proof. Since we are only concerned with finite constraints and since dominance is a partial order on nodes the set $\{\alpha(X)|X \in F\}$ contains minimal elements, i.e. nodes which are not dominated by other elements of this set. The proof that it has a unique minimal element is by contradiction.

Let X, Y be variables in F, $\alpha(X) \neq \alpha(Y)$, and let them both be minimal elements of $\{\alpha(X)|X \in F\}$. Then by Lemma 5 the only way they can be connected is in the following fashion: there is a $W \in F$ such that $X \perp Y$ at $\{W\}$. This leads to a contradiction, since it follows that $W \triangleleft^* X \wedge W \triangleleft^* Y$ although X and Y were assumed to be minimal elements. □

Lemma 7. Let φ be a normal constraint, let F be a fragment in φ, and let X, Y be different leaves of F. Then there is a variable $W \in F$ such that

$$\varphi \models X \perp Y \text{ at } \{W\}.$$

Proof. By definition F contains no variables Z_1, Z_2 such that there are labeling constraints $X{:}f(\ldots Z_1 \ldots), Y{:}f(\ldots Z_2 \ldots) \in \varphi$. This means that X and Y cannot dominate each other. But since they must be connected, we know from Lemma 5 that $X \perp Y$ at $\{W\}$ where $W \in F$ must hold. □

A.2 Properties of Relaxation and Projection

Lemma 9. The relaxation $\mathcal{R}_X(\varphi)$ of a normal constraint φ at X is normal.

Proof. By contradiction. Assume $\mathcal{R}_X(\varphi)$ is not normal, while φ is. This means that there must be two distinct occurrences of labeling constraints $Y{:}f(Y_1, \ldots, Y_n)$ and $Z{:}g(Z_1, \ldots, Z_m)$ in $\mathcal{R}_X(\varphi)$, such that $Y{\neq}Z$ *in* φ is not true. We can distinguish three cases of how Y and Z relate to the relaxation gap. Note that according to the definition of relaxation neither Y nor Z can be X^u. That leaves the cases of (a) them both being different from X^l, (b) them both being X^l, and (c) one of them being X^l and the other one being different from X^l.

(a) Since both Y and Z are different from X^l there must be labeling constraints $Y{:}f(Y'_1, \ldots, Y'_n)$ and $Z{:}g(Z'_1, \ldots, Z'_m)$ in φ, where $Y'_i = Y_i$ or

$Y_i' = X$ for all $1 \leq i \leq n$ and $Z_j' = Z_j$ or $Z_j' = X$ for all $1 \leq j \leq m$. Furthermore we know $Y{\neq}Z$ *in* φ, because φ is normal. But by the definition of relaxation this entails that $Y{\neq}Z$ *in* $\mathcal{R}_X(\varphi)$, which is in contradiction to our assumption.

(b) Exactly one of Y and Z is X^l. Without loss of generality we can assume that Y is X^l. So there must be labeling constraints $X{:}f(Y_1, \ldots, Y_n)$ and $Z{:}g(Z_1', \ldots, Z_m')$ in φ, where again $Z_j' = Z_j$ or $Z_j' = X$ for all $1 \leq j \leq m$. Since φ is normal we can conclude that $X{\neq}Z$ *in* φ. But this leads to a contradiction, because it means that by the definition of relaxation $X{\neq}Z$ *in* $\mathcal{R}_X(\varphi)$ must also be true.

(c) Both Y and Z are X^l. If $f = g$ everything is okay, because this allowed by our definition of normal constraints. Otherwise, we know that the two labeling constraints $X{:}f(Y_1, \ldots, Y_n)$ and $X{:}g(Z_1, \ldots, Z_m)$ are in φ. Because of the normality of φ we also know that $X{\neq}X$ *in* φ (note, that, furthermore, this constraint is not satisfiable). As before we can conclude from this and the definition of relaxation that $X^l{\neq}X^l$ must be in $\mathcal{R}_X(\varphi)$ and get a contradiction.

□

Before giving the proof for Lemma 12 we will show the following result, which will turn out be helpful not only for this proof.

Lemma 23. *Let* $\mathcal{M} = (\Delta, \sigma, \lambda)$ *be a* λ*-structure and* $\mathcal{M}_{\pi_l}^{\pi_u} = (\Delta_{\pi_l}^{\pi_u}, \sigma', \lambda')$ *its projection at nodes* π_u *and* π_l*. Then the following holds:*

1. $p(\pi_1){\lhd^*}p(\pi_2)$ *iff* $\pi_1{\lhd^*}\pi_2$*, for* $\pi_1, \pi_2 \in \Delta_{\pi_l}^{\pi_u}$.
2. $p(\pi_1){\neq}p(\pi_2)$ *iff* $\pi_1{\neq}\pi_2$*, for* $\pi_1, \pi_2 \in \Delta_{\pi_l}^{\pi_u}$.
3. $\sigma'(p(\pi)) = f$ *iff* $\sigma(\pi) = f$*, for* $\pi \in \Delta_{\pi_l}^{\pi_u}$.
4. $p(\pi_i) = p(\pi)i$ *iff* $\pi_i = \pi i$*, for* $\pi, \pi_i \in \Delta_{\pi_l}^{\pi_u}$.
5. $p(\pi_1) \perp p(\pi_2)$ at $p(\pi_0)$ *iff* $\pi_1 \perp \pi_2$ at π_0*, for* $\pi_0, \pi_1, \pi_2 \in \Delta_{\pi_l}^{\pi_u}$.
6. $\lambda'(p(\pi_1)) = p(\pi_2)$ *iff* $\lambda(\pi_1) = \pi_2$*, for* $\pi_1, \pi_2 \in \Delta_{\pi_l}^{\pi_u}$.

Proof.

There are two possibilities of how a projection $p(\pi)$ can relate to π.

(I) $p(\pi) = \pi$ and not $\pi_u \lhd^* \pi$.

(II) $\pi = \pi_l \pi'$ and $p(\pi) = \pi_u \pi'$ for some π'. Then $\pi_l \lhd^* \pi$.

1. (a) π_1 and π_2 are both as in (I). Hence, $p(\pi_1) = \pi_1$ and $p(\pi_2) = \pi_2$. The claim follows immediately.

   (b) π_1 is as in (I) and π_2 as in (II). Hence, $p(\pi_1) = \pi_1$ and there is a π' such that $\pi_2 = \pi_l \pi'$ and $p(\pi_2) = \pi_u \pi'$.
   $\underline{\Rightarrow}$ $p(\pi_1) \lhd^* p(\pi_2) \Rightarrow \pi_1 \lhd^* \pi_u \pi'$. But $\pi_u \pi' \lhd^* \pi_l \pi'$ and $\pi_l \pi' = \pi_2$. Therefore, $\pi_1 \lhd^* \pi_2$.
   $\underline{\Leftarrow}$ $\pi_1 \lhd^* \pi_2 \Rightarrow \pi_1 \lhd^* \pi_l \pi' \Rightarrow \pi_1 \lhd^* \pi_u \pi'$, i.e. $p(\pi_1) \lhd^* p(\pi_2)$.

   (c) The case where π_2 is as in (I) and π_1 as in (II) is not possible.

   (d) π_1 and π_2 are both as in (II). Hence, there are π' and π'' such that $\pi_1 = \pi_l \pi'$ and $p(\pi_1) = \pi_u \pi'$ and $\pi_2 = \pi_l \pi''$ and $p(\pi_2) = \pi_u \pi''$.
   $\underline{\Rightarrow}$ $\pi_u \pi' \lhd^* \pi_u \pi''$. Therefore $\pi_l \pi' \lhd^* \pi_l \pi''$ must also hold.
   $\underline{\Leftarrow}$ $\pi_l \pi' \lhd^* \pi_l \pi''$. Therefore $\pi_u \pi' \lhd^* \pi_u \pi''$ must also hold.

2. p is one-to-one.

3. $\sigma'(p(\pi)) = \sigma(\pi)$ holds by definition for any $\pi \in \Delta^{\pi_u}_{\pi_l}$.

4. (a) Both π and π_i are as in (I). The claim follows immediately.

   (b) π is as in (I) and π_i as in (II). Note that then $\pi_u = \pi_l = \pi_i$. The claim follows immediately.

   (c) π is as in (II) and π_i as in (I) is not possible.

   (d) Both π and π_i are as in (II). Hence, there are π' and π'' such that $\pi = \pi_l \pi'$ and $p(\pi) = \pi_u \pi'$ and $\pi_i = \pi_l \pi''$ and $p(\pi_i) = \pi_u \pi''$.
   $\underline{\Rightarrow}$ $\pi_u \pi'' = \pi_u \pi' i$. Hence, $\pi_l \pi'' = \pi_l \pi' i$. $\underline{\Leftarrow}$ $\pi_l \pi'' = \pi_l \pi' i$. Hence, $\pi_u \pi'' = \pi_u \pi' i$.

5. (a) π_1 and π_2 are both as in (I). It follows that π_0 must also be as in (I). Therefore, $p(\pi_1)) = \pi_1$, $p(\pi_2)) = \pi_2$ and $p(\pi_0)) = \pi_0$ and the claim immediately follows.

(b) One of π_1 and π_2 is as in (I), the other one as i (II). Without loss of generality we can assume that π_1 is as in (I). So there is a π' such that $\pi_2 = \pi_l\pi'$. $\underline{\Rightarrow}\ \pi_1 \perp \pi_u\pi'$ at $p(\pi_0)$. We can conclude that $p(\pi_0) = \pi_0$ and, since $\pi_u \lhd^* \pi_l$, $\pi_1 \perp \pi_l\pi'$ at π_0. $\underline{\Leftarrow}\ \pi_1 \perp \pi_l\pi'$ at π_0. Again we can conclude that $p(\pi_0) = \pi_0$, because $\pi_1 \perp \pi_u$ at π_0. Therefore, $p(\pi_1) \perp \pi_u\pi'$ at $p(\pi_0)$.

(c) π_1 and π_2 are both as in (II). Therefore there are π' and π'' such that $\pi_1 = \pi_l\pi'$ and $\pi_2 = \pi_l\pi''$. $\underline{\Rightarrow}\ \pi_u\pi' \perp \pi_u\pi''$ at $p(\pi_0)$. Hence, there is a π''' such that $p(\pi_0) = \pi_u\pi'''$ and π''' is a prefix of both π' and π''. This let's us conclude that $\pi_0 = \pi_l\pi'''$ and that $\pi_l\pi' \perp \pi_l\pi''$ at $\pi_l\pi'''$. $\underline{\Leftarrow}\ \pi_l\pi' \perp \pi_l\pi''$ at $\pi_l\pi'''$ and by the same argumentation as above $\pi_u\pi' \perp \pi_u\pi''$ at $p(\pi_0)$.

6. By the definition of λ'.

□

Now, we give the proof for Lemma 12 restated here.

Lemma 12. An intended solution of $\mathcal{R}_X(\varphi)$ is indeed a solution of $\mathcal{R}_X(\varphi)$.

Proof. Let $(\mathcal{M}, \alpha)$, where $\mathcal{M} = (\Delta, \sigma\lambda)$, be an intended solution of $\mathcal{R}_X(\varphi)$. By definition the projection α' of α at nodes $\alpha(X^u)$ and $\alpha(X^l)$ is defined and $(\mathcal{M}^{\alpha(X^u)}_{\alpha(X^l)}, \alpha' \cup \{Y \to p(\alpha(X^l)) \mid Y \in \mathsf{eq}_\varphi(X)\})$ satisfies φ. We have to show that $(\mathcal{M}, \alpha)$ satisfies $\mathcal{R}_X(\varphi)$, i.e. every atomic of $\mathcal{R}_X(\varphi)$ has to be satisfied by $(\mathcal{M}, \alpha)$. Let ψ be an atomic constraint in $\mathcal{R}_X(\varphi)$. Then we can distinguish the following cases.

1. ψ is of the form $Y \lhd^* Z$.
2. ψ is of the form $Y \neq Z$.
3. ψ is of the form $Y{:}f(Z_1, \ldots, Z_n)$.
4. ψ is of the form $Y_1 \perp Y_2$ at Z.
5. ψ is of the form $\lambda(Y) = Z$.

Since the proving satisfiedness of ψ works in the same way for all of the cases, we will just give the proof for the first one.

Let ψ be of the form $Y \lhd^* Z$. Then either $Y \lhd^* Z$ *in* φ or $Y' \lhd^* Z$ *in* φ, where $Y' \in \mathsf{eq}_\varphi(X)$ and $Y = X^l$, or $Y \lhd^* Z'$ *in* φ, where $Z' \in \mathsf{eq}_\varphi(X)$ and $Z = X^u$. In the first case, $\alpha'(Y) \lhd^* \alpha'(Z)$. By definition $\alpha'(Y) = p(\alpha(Y))$ and $\alpha'(Z) = p(\alpha(Z))$. So, $p(\alpha(Y)) \lhd^* p(\alpha(Z))$ and by Lemma 23 $\alpha(Y) \lhd^* \alpha(Z)$. In the second case, $p(\alpha(X^l)) \lhd^* \alpha'(Z)$. So, $p(\alpha(X^l)) \lhd^* p(\alpha(Z))$ and therefore $\alpha(X^l) \lhd^* \alpha(Z)$. Finally, in the third case, $p(\alpha(Y)) \lhd^* p(\alpha(X^l))$. Since $p(\alpha(X^u)) = p(\alpha(X^l))$, $p(\alpha(Y)) \lhd^* p(\alpha(X^u))$ and by Lemma 23 $\alpha(Y) \lhd^* \alpha(X^u)$.

□

A.3 The Safety Criterion

Proposition 14 (Safety Criterion). Let φ be a constraint, X a variable in φ, and φ' the relaxation of φ at X. Then φ is safe for relaxation at X if the following entailment is true:

$$\varphi' \models \bigwedge_{Y \in \mathcal{V}(\varphi) \setminus \{X\}} (Y \lhd^* X^u \vee Y \perp X^u \vee X^l \lhd^* Y).$$

Proof. Assume that the entailment is true; we will show that φ is safe at X. To this end, we pick a solution $(\mathcal{M}', \alpha')$ $(\mathcal{M} = (\Delta', \sigma', \lambda'))$of φ' and show that it is intended. By assumption, $(\mathcal{M}', \alpha')$ satisfied $\bigwedge_{Y \in \mathcal{V}(\varphi) \setminus \{X\}} (Y \lhd^* X^u \vee Y \perp X^u \vee X^l \lhd^* Y)$.

To proof intendedness we have to show the following two properties:

1. the projection α of α' at nodes $\alpha'(X^u), \alpha'(X^l)$ is defined.
2. $(\mathcal{M}, \alpha \cup \{Y \to p(\alpha'(X^l)) \mid Y \in \mathsf{eq}_\varphi(X)\})$, where $\mathcal{M} = (\Delta, \sigma, \lambda)$ is the projection of $\mathcal{M}'$ at nodes $\alpha'(X^u), \alpha'(X^l)$, satisfies φ.

1. To prove that property 1 holds, we have to show that $\alpha'(Y) \in \Delta^{\alpha(X^u)}_{\alpha(X_l)} \cup \{\alpha(X^u)\}$ for all $Y \in \mathcal{V}(\varphi')$. We do so by contradiction.

   Let $Y \in \mathcal{V}(\varphi')$ such that $\alpha(Y) \notin \Delta^{\alpha(X^u)}_{\alpha(X_l)} \cup \{\alpha(X^u)\}$. Then $\alpha(X^u)$ must be a prefix of $\alpha(Y)$, but they may not be equal, and $\alpha(X^l)$ must not

be a prefix of $\alpha(Y)$, which is in contradiction to the assumption that the the entailment is true.

2. To show that property 2 also holds, we first have to make sure that $\alpha \cup \{Y \rightarrow p(\alpha'(X^l)) \mid Y \in \mathsf{eq}_\varphi(X)\}$ is defined on all variables of φ. It can easily be seen that this is the case, because α is defined on all variables of φ' and the only variables of φ which are not also variables of φ' are those in $\mathsf{eq}_\varphi(X)$. Then we have to show that all atomic constraints in φ are satisfied. This is where Lemma 23 comes in handy again. Similar as in the proof for Lemma 12 we will just demonstrate how the proof works by spelling it out for one type if atomic constraint. This time we will pick the labeling constraint, i.e. we will show that $(\mathcal{M}, \alpha \cup \{Y \rightarrow p(\alpha'(X^l)) \mid Y \in \mathsf{eq}_\varphi(X)\})$ satisfies any constraint of the form $Y{:}f(Y_1, \ldots, \mathcal{Y}_n)$ in φ. We have to distinguish three cases:

   (a) $Y, Y_1, \ldots, Y_n \notin \mathsf{eq}_\varphi(X)$ and therefore $\alpha(Y) = p(\alpha'(Y))$ and $\alpha(Y_i) = p(\alpha'(Y_i))$ for $1 \leq i \leq n$. Furthermore this implies that $Y{:}f(Y_1, \ldots, \mathcal{Y}_n)$ *in* φ' and is satisfied by $(\mathcal{M}', \alpha')$. So we know that $\sigma'(\alpha'(Y)) = f$ and by Lemma 23 $\sigma(p(\alpha'(Y)) = f$. We also know that $\alpha'(Y_i) = \alpha'(Y)i$ for $1 \leq i \leq n$, and by Lemma 23 $p(\alpha'(Y_i)) = p(\alpha'(Y))i$.

   (b) $Y \in \mathsf{eq}_\varphi(X)$ and $Y_1, \ldots, Y_n \notin \mathsf{eq}_\varphi(X)$. So now $\alpha(Y) = p(\alpha'(X^l))$ and $X^l{:}f(Y_1, \ldots, \mathcal{Y}_n)$ *in* φ'. Again, we can conclude that $\sigma(p(\alpha'(X^l)) = f$ from $\sigma'(\alpha'(X^l))$ and that $p(\alpha'(Y_i)) = p(\alpha'(X^l))i$ for $1 \leq i \leq n$ from $\alpha'(Y_i) = \alpha'(X^l)i$ by Lemma 23.

   (b) There is a $j, 1 \leq j \leq n$, $Y_j \in \mathsf{eq}_\varphi(X)$ and $Y, Y_i \notin \mathsf{eq}_\varphi(X)$ for $i \neq j$ and $1 \leq i \leq n$. This means that $\alpha(Y_j) = p(\alpha'(X^l)) = \alpha'(X^u)$ and $Y{:}f(Y_1, \ldots, X^u, \ldots, \mathcal{Y}_n)$ *in* φ'. $\alpha'(X^u) = \alpha'(Y)i$. This means that not $\alpha'(X^u) \lhd^* \alpha'(Y)$ and therefore $p(\alpha'(Y)) = \alpha'(Y)$. Hence, $p(\alpha'(Y))i = \alpha(Y_j)$.

□

A.4 Properties of Chains and Cascades of Fragments

Propositions 16 and 17 stated some properties of chains of fragments that turned out to be very useful for us. Before turning to their proofs, we will show the following "quasi-transitivity" result about disjointness, which will be helpful.

Lemma 24. *If $\mathcal{V}$ and $\mathcal{W}$ are sets of variables, the following entailment is true:*

$$X \perp Y \text{ at } \mathcal{V} \wedge Y \perp Z \text{ at } \mathcal{W} \wedge \bigwedge_{\substack{V \in \mathcal{V} \\ W \in \mathcal{W}}} V \neq W \models X \perp Z \text{ at } \mathcal{V} \cup \mathcal{W}$$

Proof. Let $(\mathcal{M}, \alpha)$ be a solution of the constraint on the left-hand side. Let

$$\pi_V \in \{\alpha(V) \mid V \in \mathcal{V}\}$$

be the branching point of $\alpha(X)$ and $\alpha(Y)$, and let π_W be the analogous branching point of $\alpha(Y)$ and $\alpha(Z)$. Then there are paths $\pi'_X, \pi'_Y, \pi'_Z, \pi''_Y$ and numbers i, j, k, l and such that $i \neq k$ and $j \neq l$ and the following relations hold:

$$\begin{array}{rclrcl} \alpha(X) & = & \pi_W i \pi'_X & \alpha(Y) & = & \pi_W k \pi'_Y \\ \alpha(Z) & = & \pi_V j \pi'_Z & \alpha(Y) & = & \pi_V l \pi''_Y \end{array}$$

As π_V and π_W are different by assumption, it follows from the relations for $\alpha(Y)$ that one must be a proper prefix of the other. Let us assume, without loss of generality, that π_V is a proper prefix of π_W. Then we can write

$$\pi_W = \pi_V \pi.$$

By substituting this into the above equations, we can conclude from the equations for $\alpha(Y)$ that the first number on the path π must be l; so $\alpha(X)$ starts with $\pi_V l$, and $\alpha(Z)$ starts with $\pi_V j$. This means that $\alpha(X)$ and $\alpha(Z)$ are disjoint, and their branching point is π_V. □

Proposition 16, repeated below, states that all connection points of a chain are pairwise disjoint.

Proposition 25. *Let $\mathcal{C} = (\mathcal{F}, \mathcal{G})$ be a chain of length n in φ, and let Z_i, Z_k $(0 \leq i < k \leq n)$ be connection points of $\mathcal{C}$. Then*

$$\varphi \models Z_i \perp Z_k \text{ at } \{V_{i+1}, \ldots, V_k\},$$

where the $V_j \in F_j$.

Proof. By induction on $k - i$.

$\underline{k - i = 1.}$ Z_i and Z_{i+1} are dominated by different leaves of F_{i+1}, which are disjoint by Lemma 7.

$\underline{k - i \to k - i + 1.}$ We know by induction hypothesis that

$$\varphi \models Z_i \perp Z_{k-1} \text{ at } \{V_{i+1}, \ldots, V_{k-1}\}.$$

Furthermore, we know from Lemma 7 that

$$\varphi \models Z_{k-1} \perp Z_k \text{ at } \{V_k\}.$$

As all the F_j are disjoint, we can conclude by Lemma 24 that

$$\varphi \models Z_i \perp Z_k \text{ at } \{V_{i+1}, \ldots, V_k\},$$

which is what we wanted to show.

□

Before proceeding to the proof of Proposition 17, we will show an additional result which we need for the proof.

Lemma 26. *Let $\mathcal{C}(\mathcal{F}, \mathcal{G})$ with $\mathcal{F} = (F_1, \ldots, F_n)$ and $\mathcal{G} = (G_1, \ldots, G_{n-1})$ be a chain in φ. Let R_i be the root of fragment F_i and let X_i and Y_i be the holes of fragment F_i that dominate the roots of fragments G_{i-1} and G_i, respectively. Then for any F_i, F_{i+k} the following holds:*

$$\varphi \models X_{i+k} \lhd^* R_i \ \vee \ Y_i \lhd^* R_{i+k} \ \vee \ R_i \perp R_{i=k}.$$

Proof. By induction over k.

$\underline{k = 1.}$ By definition $X_{i+k} \lhd^* Z_i$ and $Y_i \lhd^* Z_i$, where Z_i is the root of fragment G_i. This implies $X_{i+k} \lhd^* Y_i \ \vee \ Y_i \lhd^* X_{i+k}$. Since fragments cannot overlap, it also holds that $X_{i+k} \lhd^* R_i \ \vee \ Y_i \lhd^* R_{i+k}$.

$\underline{k \to k+1.}$ The induction hypothesis gives us that $X_{i+k-1} \lhd^* R_i \vee Y_i \lhd^* R_{i+k-1}$ and $Y_{i+k-1} \lhd^* R_{i+k} \ \vee \ X_{i+k} \lhd^* R_{i+k-1}$ hold. We will make a case distinction according to which of the disjuncts are true:

1. $X_{i+k-1} \lhd^* R_i$ and $Y_{i+k-1} \lhd^* R_{i+k}$ are true. Since $X_{i+k-1} \perp Y_{i+k-1}$, $R_i \perp R_{i+k}$ must also be true.
2. $X_{i+k-1} \lhd^* R_i$ and $X_{i+k} \lhd^* R_{i+k-1}$. $R_{i+k-1} \lhd^* X_{i+k-1}$ and therefore $X_{i+k} \lhd^* R_i$.
3. $Y_i \lhd^* R_{i+k-1}$ and $Y_{i+k-1} \lhd^* R_{i+k}$. $R_{i+k-1} \lhd^* Y_{i+k-1}$ and therefore $Y_i \lhd^* R_{i+k}$.
4. $Y_i \lhd^* R_{i+k-1}$ and $X_{i+k} \lhd^* R_{i+k-1}$. Fragments cannot overlap. This gives us that (a) $Y_i \neq X_{i+k}$ and that (b) if $Y_i \lhd^* X_{i+k}$, then $Y_i \lhd^* R_{i+k}$ and if $X_{i+k} \lhd^* Y_i$, then $X_{i+k} \lhd^* R_i$. This means that $Y_i \lhd^* R_{i+k} \ \vee \ X_{i+k} \lhd^* R_i$ holds.

□

Finally, chains have the property that variables of upper fragments are either disjoint to connection points or dominate them as was stated by Proposition 17.

Proposition 17. Let $\mathcal{C} = (\mathcal{F}, \mathcal{G})$ be a chain in φ, let $F \in \mathcal{F}$ and $G \in \mathcal{G}$, and let Z be the root of G. Then for all $X \in F$,

$$\varphi \models X \perp Z \vee X \lhd^* Z.$$

Proof. Let i be the index of F in $\mathcal{F}$, and let k be the index of G in $\mathcal{G}$. The claim follows immediately for $k \in \{i-1, i\}$.

For $k \notin \{i-1, i\}$, it can be shown by contradiction: We assume that a pair $(\mathcal{M}, \alpha)$ satisfies both φ and $Z \lhd^* X$. Under this assumption, we distinguish cases as to whether $\alpha(Z) \neq \alpha(X)$ or $\alpha(Z) = \alpha(X)$.

$\underline{\alpha(Z) = \alpha(X)}$. We can further distinguish whether i is larger or smaller than k. The two cases can be handled analogously, so we will only consider the case $i < k$ here. Let X_k be the hole of fragment F_k that dominates the root of fragment G_k and Y_k the hole that dominates the root of G_{k-1}. Since fragments cannot overlap, the root of F_k dominates the root of F in $(\mathcal{M}, \alpha)$. Then, by Lemma 26, X_k also dominates the root of F, which contradicts the disjointness of leaves of fragments.

$\underline{\alpha(Z) \neq \alpha(X)}$. Again we will only consider the case where $i < k$. Let Y_k and X_k be as before, let Y_i be defined analogously to Y_k, and let R and R_k be the roots of fragments F and F_k respectively. Since $\alpha(Z) \triangleleft^* \alpha(X)$ and $\alpha(Y_k) \triangleleft^* \alpha(Z)$, $\alpha(Y_k) \triangleleft^* \alpha(X)$ also holds. This means that neither $Y_i \triangleleft^* R$ nor $R \perp R_k$ is true. In this case $X_k \triangleleft^* R$ must be satisfied according to Lemma 26, which is in contradiction to the disjointness of the leaves of fragment F_k.

□

Now, we will turn to the proof of Proposition 21, which states some properties of cascades of fragments.

Proposition 27. *Let $\mathcal{CF} = (S_1, \ldots, S_n)$ be a cascade of fragments in φ, let $R_1, \ldots, R_n$ be the roots of $S_1, \ldots, S_n$, and let $U_0, \ldots, U_{n-1}$ be the anchoring points of $\mathcal{CF}$. Then*

$$\varphi \models U_i \triangleleft^* X$$

for any $X \in S_{i+1} \cup \ldots \cup S_n$ and

$$\varphi \models X \triangleleft^* U_i \vee X \perp U_i$$

for any $X \in S_1 \cup \ldots \mathcal{S}_i$.

Proof. By induction over n, the length of the cascade.

$\underline{n = 1..}$ There is only one fragment. U_0 is the root of this fragment and U_1 a leaf.

$\underline{n \to n+1.}$ Let $\mathcal{CF} = (S_1, \ldots, S_{n+1})$ be a cascade of length $n+1$ with anchoring points $U_0, \ldots, U_n$. Let $\mathcal{CF}' = (S_2, \ldots, S_{n+1})$. $\mathcal{CF}'$ is a cascade of length n and the claim holds for $\mathcal{CF}'$ by the induction hypothesis. In particular, for all $X \in S_2 \cup \ldots \cup S_{n+1}$ $U_1' \lhd^* X$, where U_1' is the root of fragment S_2. By definition $U_1 \lhd^* U_1'$. Hence, $U_1 \lhd^* X$ for all $X \in S_2 \cup \ldots \cup S_{n+1}$, which proves the first part of the claim. U_1 is a leaf of fragment S_1. Therefore all variables of S_1 are disjoint to U_1 or dominate it, i.e. $X \lhd^* U_1 \vee X \perp U_1$ for all $X \in S_1$. By the first part of the claim we know that U_1 dominates any U_i with $1 < i \leq n$. Hence, all variables of S_1 are either disjoint to U_i or dominate it. This completes the proof for the second part of the claim.

□

A.5 A Safe Syntax/Semantics Interface

In this Section we will give the proof for Lemmas 19 and 22, which claim that the semantic contribution of a noun phrase has the structure of a chain and while the semantic contribution of a verb phrase is a cascade of fragments to which a chain may be attached.

Lemma 19. Let T be a parse tree of our grammar, and let φ be the constraint that our syntax/semantics interface assigns to it. Furthermore, let t be a subtree of T whose root is labeled with `NP`, let n be the number of `NP` nodes in t, and let φ_t be the conjunction of the constraints corresponding to the nodes of t. Then there is a chain $\mathcal{C} = ((F_1, \ldots, F_n), (G_1, \ldots, \mathcal{G}_{n-1}))$ of length n covering φ:

$$\mathcal{V}(\varphi_t) = \bigcup_{F \in \mathcal{F}} F \cup \bigcup_{G \in \mathcal{G}} G.$$

Proof. By induction over n.

$\underline{n = 1.}$ In this case, t has the form $\texttt{NP}(\texttt{Det}(W_1)\ \overline{\texttt{N}}(\texttt{N}(W_2)))$, where W_1 and W_2 are words. As we can easily see from the rules (b2) and (b3), the resulting constraint φ_t is a single fragment, so $\mathcal{C} = ((\mathcal{V}(\varphi_t)), ())$ is a chain of length 1.

$\underline{n-1 \to n.}$ Let t' be the largest proper subtree of t whose root is labeled with NP; then $t = \texttt{NP}(\texttt{N}(W_1)\ \texttt{PP}(\texttt{P}(W_2)\ t'))$, where W_1 and W_2 are again words, and t' contains $n-1$ NP nodes.

By induction hypothesis, the semantics $\varphi_{t'}$ of t' is a chain $\mathcal{C}' = (\mathcal{F}', \mathcal{G}')$ of length $n-1$ such that the fragments in $\mathcal{F}'$ and $\mathcal{G}'$ form a partition of $\mathcal{V}(\varphi_{t'})$; say, $\mathcal{F}' = (F'_1, \ldots, F'_{n-1})$ and $\mathcal{G}' = (G'_1, \ldots, G'_{n-2})$.

According to the syntax/semantics interface, we obtain φ_t from $\varphi_{t'}$ by applications of the rules (b2), (b4), and (b5). These rules introduce two new fragments; let F_0 be the fragment consisting of the contributions of rules (b2) and (b4), and let G_0 be the fragment introduced by rule (b5). Furthermore, rule (b5) extends fragment F'_1 to a fragment F_1 which contains a new leaf, namely the scope. Finally, there are dominance constraints, demanding that the root of G_0 be dominated by the new leaf of fragment F_1 as well as a leaf of fragment F_0.

This means that $(F_0, F_1, F'_2, \ldots, F'_n)$ and $(G_0, G'_1, \ldots, G'_{n-1})$ form a chain in φ. Its length is n, and as it contains all variables that the three rule applications introduced, its fragments contain all variables in φ_t. □

Lemma 22. Let T be a parse tree of our grammar and let φ be the constraint that our syntax/semantics interface assigns to it. Furthermore, let t be a subtree of T whose root node ν is labeled with VP, let n be the number of verb nodes, i.e. IV, TV, and CV nodes, in t, and let φ_t be the conjunction of the constraints corresponding to the nodes of t. Then there are two possibilities for what the structure of φ_t may look like.

1. There is a cascade of fragments $(S_1, \ldots, S_n)$ with anchoring points $U_0, \ldots, U_n$, where $n \geq 1$ is the number of verb nodes in t. X_ν (which is the variable in the semantic representation associated with the node ν of the parse tree, i.e. the root of t) is the root of fragment S_1. This is depicted in Figure 5.12 (a).

2. There is a cascade of fragments $(S_1, \ldots, S_n)$ with anchoring points $U_0, \ldots, U_n$, where $n \geq 1$ is the number of verb nodes in t. X_ν is the root of fragment S_1. Furthermore, there is a chain of fragments $((F_1, \ldots, F_m), (G_1, \ldots, \mathcal{G}_{m-1}))$, $m \geq 1$, and the root of fragment S_n is dominated by a hole of fragment F_1. This is depicted in Figure 5.12 (b).

No variable of φ_t takes part in more than one of these fragments and all variables except for $X_1^l, \ldots, X_n^l$ take part in at least one of the fragments. X_i^l is dominated by a leaf of fragment S_i which is disjoint to U_i.

Proof. By induction over n.

$\underline{n = 1..}$ t will either look like $\texttt{VP}(\texttt{IV}(W))$ or like $\texttt{VP}(\texttt{TV}(W)\texttt{NP}(t')$, where W is a word. The semantic representation for the first case is derived by rules (b7) and (b10") of the syntax/semantics interface. It is easy to see that the contribution is as in case 1.) of the lemma, where the cascade is of length 1. The contribution of t' is a chain as described in Lemma 19. It is connected to the semantics of the verb by rules (b8) and (b10"). Again it is easy to see that the result is as in case 2.) of the lemma and the cascade is of length 1.

$\underline{n \to n+1..}$ There is only one possibility of embedding verb nodes, namely the following rule: $\texttt{VP} \to \texttt{CV VP}$. Hence, the parse tree for t looks like this $\texttt{VP}(\texttt{CV}(W)\texttt{VP}(t'))$, where t' contains n verb nodes. By the induction hypothesis the structure of t' can be either as in case 1.) of the lemma or as in case 2.).

$\underline{\text{case 1..}}$ Let S_0 be the fragment introduced by rule (b9) and R its root. A leaf of this fragment dominates the root of fragment S_1. Let X be this leaf. Then the contribution of t is a cascade $\mathcal{CF}' = (S_0, S-1, \ldots, S_n)$ with anchoring points $R, X, U_1, \ldots, U_n$.

$\underline{\text{case 2..}}$ In this case the semantic contribution of t' consists of a chain and a cascade. Rule (b9) only affects the cascade. As in case 1.) it adds a fragment to the top, so that the result is still a chain and a (somewhat enlargened) cascade.

□

Bibliography

Bartsch, R. (1984). The Structure of Word Meanings: Polysemy, Metaphor, Metonymy. In F. Landman and F. Veltman (Eds.), *Varieties of Formal Semantics*, Groningen-Amsterdam Studies in Semantics. FORIS Publications.

Bos, J. (1996). Predicate Logic Unplugged. In *Proceedings of the 10th Amsterdam Colloquium*.

Copestake, A. and T. Briscoe (1995). Semi-productive polysemy and sense extension. *Journal of Semantics*.

Copestake, A. and A. Lascarides (1998). Pragmatics and Word Meaning. *Journal of Linguistics*.

Copestake, A., A. Sanfilippo, T. Briscoe, and V. de Paiva (1993). *Inheritance, Defaults, and the Lexicon*, Chapter The AQUILEX LKB: An Introduction. Camebridge University Press.

Cruse, D. A. (1991). *Lexical Semantics*. Cambridge Textbooks in Linguistics. Cambridge: Cambridge University Press.

Dölling, J. (1994). Sortale Selektionsbeschränkungen und systematische Bedeutungsvariationen. In *Kognitive Semantik. Ergebnisse, Probleme, Perspektiven*, Tübinger Beiträge zur Linguistik 395. Tübingen: Schwarz, M.

Dölling, J. (1997). Semantic Form and Abductive Fixation of Parameters. In *From Underspecification to Interpretation*, Working Papers of the Institute for Logic and Linguistics Nr. 29. Heidelberg, Germany: van der Sandt, R. and Blutner, R. and Bierwisch, M.

Egg, M. (1999). Reinterpretation from a synchronic and a diachronic point of view. Technical report, Arbeitspapiere der FG Sprachwissenschaft, Universität Konstanz.

Egg, M., J. Niehren, P. Ruhrberg, and F. Xu (1998, August). Constraints over Lambda-Structures in Semantic Underspecification. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'98)*, Montreal, Canada, pp. 353–359.

Fries, N. (1980). *Ambiguität und Vagheit*. Linguistische Arbeiten 84. Tübingen: Max Niemeyer Verlag.

Hobbs, J. R., M. E. Stickel, D. E. Appelt, and P. Martin (1993). Interpretation as abduction. *Artificial Intelligence Vol. 63 No. 1-2*.

Jaspers, J. and A. Koller (1999). A Calculus for Direct Deduction with Dominance Constraints. In *Proceedings of the Twelfth Amsterdam Colloquium*, Amsterdam.

Koller, A. (1999). Constraint Languages for Semantic Underspecification. Master's thesis, Universität des Saarlandes.

Nunberg, G. (1995). Transfers of Meaning. *Journal of Semantics 12*, 109–132.

Pinkal, M. (1995). Radical Underspecification. In P. Dekker and M. Stokhof (Eds.), *Proceedings of the 10th Amsterdam Colloquium*. University of Amsterdam.

Pustejovsky, J. (1994). *Computational Lexical Semantics*, Chapter Linguistic Constraints on Type Coercion. Cambridge University Press.

Pustejovsky, J. (1995). *The Generative Lexicon*. Cambridge: MIT Press.

Reyle, U. (1993). Dealing with Ambiguities by Underspecification: Construction, Representation, and Inference. *Journal of Semantics*.

van Deemter, K. and S. Peters (1996). *Semantic Ambiguity and Underspecification*. Stanford: CSLI.

Verspoor, C. (1997). *Contextually-Dependent Lexical Semantics*. Ph. D. thesis, University of Edinburgh.

Xu, F. (1998). Underspecified Representation and Resolution of Ellipsis. Master's thesis, Universität des Saarlandes.