A General Reasoning Scheme for Underspecified Representations

Esther König, Uwe Reyle
University of Stuttgart, Institute for Computational Linguistics
Azenbergstr. 12, D-70174 Stuttgart, http://www.ims.uni-stuttgart.de

Abstract

In this paper we present an *underspecified logic*, i.e. a pair consisting of a proper underspecified semantic representation formalism and a deductive component that directly operates on these structures. We show how the main features of this formalism can be imported into other existing underspecified semantic representation formalisms. We also show that deduction rules may be imported there along the same lines. The set of importable rules will of course depend on the completeness properties of the particular formalisms.

The current paper is a short version of [10].

1 The landscape of Underspecified Semantic Representations

Underspecified semantic representations have attracted increasing interest within computational linguistics. Several formalisms have been developed that allow to represent sentence or text meanings with that degree of specificity that is determined by the context of interpretation. As the context changes they must allow for (partial) disambiguation steps performed by a process of refinement that goes hand in hand with the construction algorithm. And as the interpretation of phrases often[1] relies on deductive principles and thus any construction algorithm must be able to integrate the results of deductive processes, any semantic formalism should be equipped with a deductive component that operates directly on its semantic forms.

We call a meaning ϕ of a representation formalism $\mathcal{L}$ *underspecified*, if it represents an ambiguous natural language sentence or text in a more compact manner than by a disjunction of all its readings. $\mathcal{L}$ is called *semantic* if its representations are model-theoretically interpretable or if it comes with a disambiguation device that turns underspecified representations into sets of model-theoretically interpretable representations.[2] If $\mathcal{L}$'s disambiguation steps produce representations of $\mathcal{L}$ only, then $\mathcal{L}$ is called *closed*. And if $\mathcal{L}$'s disambiguation device produces all possible refinements of any ϕ, then $\mathcal{L}$ is called *complete*. Completeness is of course dependent on the particular natural language fragment $\mathcal{L}$ is supposed to cover. In this paper we restrict ourselves to the fragment of simple sentences containing singular indefinite as well as quantified NPs,[3] relative clauses and negation. To give an example what completeness involves let us consider a sentence with three quantified NPs with underspecifed scoping relations. Then $\mathcal{L}$ must be able to represent all $2^{3!} = 64$ refinements, i.e. partial and complete disambiguations of this sentence. For many formalisms the question whether they are complete wrt. to a particular fragment, or not, is not decided yet. We, therefore, take a very liberal view and interpret 'complete' more in the sense of 'intended to be comlete' than in the sense of a precise characterization of expressive power. A formalism $\mathcal{L}$ is called *proper* if it is closed and complete. It is *c-deductive* (or 'classically deductive') if there is an inference mechanism for the disjunction of fully specified formulas the underspecified formula is supposed to represent. The formalism is called *u-deductive* if it is equipped with a deductive component that operates directly on the underspecified forms.

[1] E.g. in order to apply nominal and temporal resolution, consistency checks, integration of world knowledge or other non-compositional interpretation principles.

[2] Note that the second disjunct requires that either the underspecified representations themselves or the disambiguation algorithm is subject to certain demands on wellformedness, as, e.g., the so-called 'free-variable constraint' ([11], [9]). Although we think that this is a very important distinction (in particular under computational aspects) we do not distinguish here between those formalisms which are provided with a check of such meta-level constraints directly for underspecified representations and those formalisms whose well-formedness test requires all the total disambiguations.

[3] With the additional assumption that the interpretation of indefinite NPs is clause-bounded.

Table 1: Comparison of various underspecified formalisms with respect to some desirable logical properties.

	LFG	**MG**	**MRS**	**QLF**	**UDRS**	**USDL**	**UL**
semantic	yes	yes	no	yes	yes	yes	yes
closed	yes	yes	yes	yes	yes	yes	yes
complete	no	no	no	almost	yes	yes	yes
proper	no	no	no	almost	yes	yes	yes
c-deductive	yes	yes	no	yes	yes	yes	yes
u-deductive	no	no	no	no	yes	no	yes
cu-deductive	no	no	no	no	no	no	yes

If the deduction on the underspecified formulas can be merged with disambiguation steps, it is named *cu-deductive*. Table 1 gives a classification of some underspecified formalisms according to these properties. **LFG** stands for the linear logic approach to LFG semantics [6]. **MG** means Montague Grammar [7]. **MRS** is the Minimal Recursion Semantics of [4]. Quasi Logical Forms **QLF** and underspecification has been explored in [2]. For Underspecified Discourse Representation Structures **UDRS** see [13]. **USDL** is one of the formalisms which have been described in the section on underspecification in [3]. **UL** is the *U(nderspecified) L(ogic)*, we present in this paper. As can be judged from the available literature, almost all formalisms are semantic. The completeness property will be discussed subsequently for each formalism. Obviously, all the 'semantic' formalisms are classically deductive, but only **UDRS**'s and **UL** are u-deductive. And only **UL** is cu-deductive.

The underspecified logic **UL** is a pair consisting of a proper underspecified semantic representation formalism $\mathcal{L}$, and a deductive component that directly operates on these structures. For the purpose of this paper and also for the sake of comparison we have split up the representations ϕ of a formalism $\mathcal{L}$ into three components, B, C, and D. M specifies the building blocks of the representation language, and C tells us how these building blocks are to be put together. D is the disambiguation device which implements the construction of the individual meaning representations from a meaning description $\langle B, C\rangle$. In the remainder of this section we present the Linear Logic approach to LFG semantics[4] from the point of view of B, C, and D. Section 2 will then explain the deductive principles of our underspecified logic **UL** and will show how these principles can be imported into LFG.

1.1 Linear Logic approach to LFG Semantics

In the case of [6]'s linear logic approach to LFG semantics M consists of linear logic formulas built up from semantic projections (i.e. formulas of the form $h_\sigma \rightsquigarrow Y$ with h referring to an f-structure and Y being a variable or a formula of higher order predicate logic). C reflects the hierarchical ordering of the underlying f-structure. The structure in (2) is the f-structure of the ambiguous sentence (1).

Every boy saw a movie. (1)

$$f:\begin{bmatrix} \text{SUBJ}\ g:\begin{bmatrix}\text{SPEC}\ \texttt{every}\\ \text{PRED}\ \texttt{boy}\end{bmatrix}\\ \text{OBJ}\quad h:\begin{bmatrix}\text{SPEC}\ \texttt{a}\\ \text{PRED}\ \texttt{movie}\end{bmatrix}\\ \text{PRED}\ \texttt{see}\end{bmatrix} \tag{2}$$

The semantic projections associate the following meaning constructors for *every boy*, *a movie* and *saw*:

[4]The other formalisms are discussed in the full version of this paper [10].

$$\left\{\begin{array}{l} \forall \textsc{g}, R.(\forall x.g_\sigma \rightsquigarrow x \multimap \textsc{g} \rightsquigarrow R(x)) \multimap \textsc{g} \rightsquigarrow \texttt{every}(\texttt{boy}, R), \\ \forall \textsc{h}, S.(\forall y.h_\sigma \rightsquigarrow y \multimap \textsc{h} \rightsquigarrow S(y)) \multimap \textsc{h} \rightsquigarrow \texttt{a}(\texttt{movie}, S), \\ \forall x, y.g_\sigma \rightsquigarrow x \otimes h_\sigma \rightsquigarrow y \multimap f_\sigma \rightsquigarrow \texttt{see}(x, y) \end{array}\right\} \quad (3)$$

D consists of a proof method for linear logic which in the case of (3) allows for two different formulas to be derived. If C only contains restrictions derived from f-structure, then the formalism is not complete. The incompleteness can be shown in a similar way as [8] does for the standard HPSG semantics (cf. [12]). E.g. for a verb with three complements SUBJect, OBJect, and IOBJect, one cannot state a constraint that SUBJ must have wide scope over OBJ while leaving open the scope relation between SUBJ and IOBJ. We must add additional constraints to the effect that certain proof steps are excluded. But this requires also that the proof theory of linear logic is made sensitive to this kind of constraint: a non-trivial extension (viz. the contribution by Crouch and van Genabith in [3]).

2 UL - Underspecified Logic

The next section presents a general formalism that subsumes the above mentioned ones. Our policy is to keep the formalism as neutral as possible with respect to the particular shape of B and C. Its design is mainly dependent on the objective of being proper, semantic and (c)u-deductive. First, the general issues of such a formalism are discussed, before defining the details of its syntax and semantics.

2.1 The Ambiguity Connective and the Consequence Relation

Suppose a hearer is confronted with an utterance that is noncommittal on some point of intense interest to him. Then he may well identify a number of alternatives to complete the utterance with the information that he considers essential. But he is unable to choose among them as long as this bit of information is not provided. On the basis of this intuition the semantic meaning of ambiguous utterances A is adequately represented by a (partial) function $[\![A]\!] : \mathcal{K} \rightarrow \{[\![A_i]\!]\}_{i=1,\ldots,n}$ from contexts $\mathcal{K}$ to the set of fully specified readings $\{[\![A_i]\!]\}_{i=1,\ldots,n}$ of A. As not all contexts do provide sufficient information to identify exactly one reading we may identify fully specified readings $[\![A_i]\!]$ with constant functions $[\![A_i]\!] : \mathcal{K} \rightarrow \{[\![A_i]\!]\}$ and generalize $[\![A]\!]$ to functions mapping contexts $\kappa \in \mathcal{K}$ to functions from contexts to meanings of less ambiguous expressions. We thus assume that the underlying logical formalism is proper. To see what syntactical devices we need to guarantee properness consider (4), (5), and (6).

James knows Jeeves. (4)

He smokes. (5)

James knows Jeeves. He smokes. (6)

Pronouns as well as proper names may have more than one possible reference, leading to ambiguities in (4), (5), and (6). The problem is that when (4) and (5) are combined to (6), their ambiguities do not multiply out. To see this suppose the domain of individuals consists of four people, $\{a, b, c, d\}$ of which $\{a, b\}$ are bearers of the name *James* and $\{c, d\}$ bear the name *Jeeves*. Then (4) and (5) are both four times ambiguous, or have four *possible* disambiguations. The sentence *He smokes* is also four times ambiguous if uttered in the context of (4), as in (6). In this context, however, the pronoun *he* is *contextually bound*, or *restricted* by the constraint that its antecedent should be either James or Jeeves. We will use a coindexing device as in (7) and (8) to indicate contextual disambiguation.

James knows Jeeves$_1$*. He*$_1$ *smokes.* (7)

James$_1$ *knows Jeeves*$_2$*. He*$_{1/2}$ *smokes.* (8)

The effect of contextual restriction on possible disambiguations is that the possible disambiguations of simple sentences do not multiply out when they are contextually combined. Taken the contextual restriction in (7) we do not get 16 readings for the whole sequence in (6), but only

four. More interesting is the contextual disambiguation in (8). Although any of $\{a, b, c, d\}$ may be the referent of the pronoun *he* we only get eight readings for (8). It is important to note that this kind of contextual restriction on possible disambiguation is at work for all kinds of ambiguities. The sentences (9), (10), and (11) are sample cases of ambiguities that do not involve quantifier scope.

I like squash. (9)

The students get £ 100. (10)

Firemen are available. (11)

The sentence *It tastes wonderful* expresses a (post-hoc partial) restriction to (9) that excludes the interpretation of *squash* as a sport. If (10) is uttered with the floating quantifier *each* then the collective reading is excluded. And the existential reading of (11) may be forced by adding an ellipsis construction like *and their chief too*[5].

As contextual disambiguation applies to all kinds of ambiguities the coindexing device must be equally flexible. Consider (12) (taken from [14]).

$$\frac{\begin{array}{c}\textit{If [the students get £100]}_i \textit{ then they buy books.} \\ \textit{[The students get £100]}_j\textit{.}\end{array}}{\textit{The students buy books.}} \tag{12}$$

According to the most natural interpretation the two occurrences of

The students get £100. (13)

are taken to mean the same thing, i.e. i is taken to be equal to j. Under this coindexing constraint the meaning of the premise of (12) is given by (15) not by (14), where A_1 represents the first and A_2 the second reading of the second sentence of (12).

$$((A_1 \to B) \vee (A_2 \to B)) \wedge (A_1 \vee A_2) \tag{14}$$

$$((A_1 \to B) \wedge A_1) \vee ((A_2 \to B) \wedge A_2) \tag{15}$$

Note that *The students buy books* and *They buy books* must also be correlated in (15). Otherwise the argument would not be sound (under the assumption that the distributive reading of buying books (we mean: distributive with respect to the set of students) is not logically equivalent to the collective reading).

Before we go on let us make a small remark on non-monotonicity. The choice a context makes among different readings may be subject to revision, as shown in (16) and (17).

James enters the room.
When he smokes Mary gets angry. (16)

James enters the room.
When he smokes Lord Leicester wants to have a brandy. (17)

Ambiguity and context change thus result in non-monotonicity. This does, however, not affect the problem of ambiguous consequence we are discussing in this section. The reason is the following: We take a set of underspecified representations to be given as the result of interpreting – say – a piece of text. In particular we assume that contextual disambiguations relevant for the understanding of the text have been made by the interpreter. That is we assume the data to be decorated with a fixed set of indices that express the contextual choices made by the interpreter. Given this kind of data, we want to know what can be derived from it.

[5]Proposals for the parallel disambiguation of quantifiers in the context of coordination and elliptical construction have been made in [1] and [5].

2.1.1 The ambiguity connective, $\sharp$

Let $\sharp$ be an operator that represents A's ambiguity between (possibly ambiguous) sentences A_1 and A_2 by $A_1 \sharp A_2$. We have seen that any attempt to represent the interpretation of $\sharp$ by a function $[\![A_1 \sharp A_2]\!]$ is doomed to failure, because its interpretation does not take contextual disambiguation into account. It must thus be parametrized by contexts κ to $[\![A_1 \sharp A_2]\!]^\kappa$. What are the properties of $[\![\cdot]\!]^\kappa$?

First of all, it has to guarantee that the $\sharp$-operator distributes over negation.[6] The ambiguity in (6) is present in exactly the same way in *James doesn't know Jeeves. He doesn't smoke.* This means that

$$[\![\neg(A_1 \sharp A_2)]\!]^\kappa = [\![(\neg A_1)\sharp(\neg A_2)]\!]^\kappa \quad \text{for any } \kappa \tag{18}$$

For conjunction and implication, $\odot \in \{\wedge, \rightarrow\}$, the case is more complicated, because they are binary and thus must respect (mutual) contextual restrictions between the formulas, A and B, they combine. If contextual constraints affect A and B, then the whole product set

$$\delta(A \odot B) := \{[\![A_1 \odot B_1]\!], [\![A_1 \odot B_2]\!], [\![A_2 \odot B_1]\!], [\![A_2 \odot B_2]\!]\} \tag{19}$$

is no longer available. The set is restricted to pairs

$$\delta^\kappa(A \odot B) := \{[\![A_{\iota_1} \odot B_{\nu 1}]\!] \ldots, [\![A_{\iota_n} \odot B_{\nu n}]\!]\} \tag{20}$$

that are admitted by the constraint set κ expressing coindexations between (sub-)phrases of A and B.[7] This means that the interpretation function $[\![\cdot]\!]^\kappa$ must satisfy the following property for two-place connectives $\odot \in \{\wedge, \rightarrow\}$.

$$[\![(A_1 \sharp A_2) \odot (B_1 \sharp B_2)]\!]^\kappa = [\![(A_{\iota_1} \odot B_{\nu 1})\sharp \ldots \sharp(A_{\iota_n} \odot B_{\nu n})]\!]^\kappa \tag{21}$$

$\delta^\kappa(A)$ is a *disambiguation operation* that respects contextual restrictions within A. We may assume that the contextual constraints, κ, are given as sets of equations, or membership relations, indicating coreferentiality of certain term expressions, or, more generally, correlatedness of phrase meanings. Consider again (7) and (8). Assume that A, B, C and D unambiguously refer to the individuals a, b, c and d, respectively. Then (22) corresponds to (7) and (23) to (8). (24) is no possible disambiguation.

$$\begin{aligned} &\delta^{\{[\![He]\!]=[\![James]\!]\}}(\textit{James knows Jeeves. He smokes.}) \\ &= \left\{ \begin{array}{l} \textit{A knows C. A smokes., B knows C. B smokes.,} \\ \textit{A knows D. A smokes., B knows D. B smokes.} \end{array} \right\} \end{aligned} \tag{22}$$

$$\begin{aligned} &\delta^{\{[\![He]\!]\in\{[\![James]\!],[\![Jeeves]\!]\}\}}(\textit{James knows Jeeves. He smokes.}) \\ &= \delta^{\{[\![He]\!]=[\![James]\!]\}}(\textit{James knows Jeeves.He smokes.}) \\ &\quad \cup \left\{ \begin{array}{l} \textit{A knows C. C smokes., B knows C. C smokes.,} \\ \textit{A knows D. D smokes., B knows D. D smokes.} \end{array} \right\} \end{aligned} \tag{23}$$

$$\begin{aligned} &\delta^{\{[\![He]\!]\in\{[\![James]\!],[\![Jeeves]\!]\}\}}(\textit{James knows Jeeves. He smokes.}) \\ &= \{\textit{A knows C. B smokes.}\} \end{aligned} \tag{24}$$

Consider again (12) the data of which are abbreviated here as (25). Let A_1 and A_2, B_1 and B_2 be the two readings of the sentences A and B, respectively. (26) makes this explicit. If we assume that the only contextual disambiguation between A and B concerns the binding of *they* by *the students* then (26) is equivalent to (27) by applying (21) to (25)'s antecedent. (27) is equivalent to (28) if we assume that no contextual disambiguation occurs between the two occurrences of A. And (27) is equivalent to (29) if we assume that the two occurrences of A are co-indexed.

[6] We restrict ourselves to cases here where the presence of negation doesn't increase the set of possible readings as, e.g. in *John doesn't admire any linguist*, which is ambiguous, whereas *John admires any linguist* is not.

[7] In the following $\wedge$ is the dynamic (left-associative) conjunction operation on formulas with the intuition that the first argument presents the context in which the second argument is asserted.

$$(A \to B) \wedge A \tag{25}$$

$$(A_1 \sharp A_2) \to (B_1 \sharp B_2)) \wedge A_1 \sharp A_2) \tag{26}$$

$$((A_1 \to B_1) \sharp (A_1 \to B_2) \sharp (A_2 \to B_1) \sharp (A_2 \to B_2)) \wedge A_1 \sharp A_2) \tag{27}$$

$$\begin{array}{l} ((A_1 \to B_1) \wedge A_1) \sharp ((A_1 \to B_1) \wedge A_2) \\ \sharp ((A_1 \to B_2) \wedge A_1) \sharp ((A_1 \to B_2) \wedge A_2) \\ \sharp ((A_2 \to B_1) \wedge A_1) \sharp ((A_2 \to B_1) \wedge A_2) \\ \sharp ((A_2 \to B_2) \wedge A_1) \sharp ((A_2 \to B_2) \wedge A_2) \end{array} \tag{28}$$

$$\begin{array}{l} ((A_1 \to B_1) \wedge A_1) \sharp ((A_1 \to B_2) \wedge A_1) \\ \sharp ((A_2 \to B_1) \wedge A_2) \sharp ((A_2 \to B_2) \wedge A_2) \end{array} \tag{29}$$

2.1.2 The consequence relation

Suppose a reasoning step is performed by an agent in order to make some information B explicit that is contained in his mental state, A, only implicitly. Then B automatically inherits all the contextual restrictions attached to the information bits it is derived from. Consider once more (12) with the coindexing constraints given in (30)

$$\frac{\begin{array}{c} \textit{If } [[\textit{the students}]_1 \textit{ get £100}]_2 \textit{ then } [[\textit{they}]_1 \textit{ buy books}]_3. \\ [[\textit{The students}]_1 \textit{ get £100}]_2. \end{array}}{[[\textit{The students}]_i \textit{ buy books}]_j.} \tag{30}$$

Given the kind of forward reasoning performed by an agent then the conclusion B in (30) must carry the same indices as the consequent of the implication, i.e. the index i of the conclusion must be set equal to 1, and j to 3. This co-indexing is an essential part of the derivation, because B is completely disambiguated by the contextual constraints imposed by what the agent knows, i.e. the data A, it is derived from. In technical terms, $|\delta^\kappa(B)| = |\delta^\kappa(A \wedge B)|$ for all κ.

A case to be distinguished from this one arises, for example, if some person, P_1, asks a question B to some other person, P_2. For reasons of dialogue cohesion there will be certain contextual restrictions between the question B and the representation A of P_2's knowledge.[8] But there may be ambiguities in the query that P_2 cannot resolve. Suppose the query B is *Do they*$_1$ *buy books?* and A corresponds to the data in (30). Here the interpretation of the pronoun *they* in B is correctly bound to the NP $[\textit{The students}]_1$ by co-indexation. Now P_2 may well – by the kind of forward reasoning described above – derive $[[\textit{they}]_1 \textit{ buy books}]_3$. But the question is, under which circumstances will he give the answer *yes* to P_1's question? There are two cases to be distinguished here. P_1 may not be aware of the ambiguity in neither $[[\textit{they}]_1 \textit{ buy books}]_3$ nor in the representation B of P_1's question. In this case he will consider the two representations equivalent and give a positive answer. So, let us assume he knows about $[[\textit{they}]_1 \textit{ buy books}]_3$'s ambiguity. Then he must conceive the possibility that the meanings of his and P_1's occurrence may diverge. This will in fact be the case if P_1 actually had a collective meaning in mind but did not make this explicit in the way he formulated the query. And if P_2 contextually disambiguates $[[\textit{they}]_1 \textit{ buy books}]_3$ to the distributive reading, because he came to learn more about the students practices, the correlation between the antecedent and consequent of the implication in (25), and the absolut amount of money the students actually get. Being aware of this possible divergence P_2 will not give the answer *yes*. If he is sure about the distributive reading he will instead give an answer that provides the disambiguating information to P_1's query, namely *They each bought books.*. And if his own representation is ambiguous as well then he may make this explicit by answering *Either they each bought books or they bought books together.* This last answer shows that P_2's representation of the ambiguous $[[\textit{they}]_1 \textit{ buy books}]_3$ is equivalent to the disjunction of its disambiguations (disjunction modulo contextual constraints within A, that

[8] Among them restrictions concerning the interpretation of proper names, pronouns, tenses and so on.

is). The first answer indicates that on the other hand P_2 represents P_1's ambiguous query as equivalent to the conjunction of its disambiguations.

Thus, if B is ambiguous between B_1 and B_2, then $B_1 \sharp B_2 \models B_1 \sharp B_2$ is true if all of $B_1 \models B_1$, $B_1 \models B_2$, $B_2 \models B_1$, and $B_2 \models B_2$ are. Only if the two occurrences of B are coindexed, i.e. if P_2 knows that his and P_1's B mean the same thing, then $B_i \models B_i$ is true if $B_1 \models B_1$ and $B_2 \models B_2$ is. Hence both scenarios we discussed, the forward reasoning and the dialogue case, may be subsumed by the following general definition of ambiguous consequence.

$$A \models^{\kappa} B \quad \text{iff} \quad \text{for all } \delta^{\kappa} \;\; \delta^{\kappa}(A) \models \delta^{\kappa}(B) \tag{31}$$

The following versions of reflexivity, monotonicity and transitivity hold for $\models^{\kappa}$.

Theorem 1

Reflexivity: $A \wedge B \models^{\kappa} B$ *iff for all* $B_i, B_j \in \delta^{\kappa}(B)$ $A \wedge B \models^{\kappa} B_i \leftrightarrow B_j$.

Monotonicity: If $A \models^{\kappa} B$ *and* $\kappa \subseteq \kappa'$, *then* $A \wedge A' \models^{\kappa'} B$.

Transitivity: If $A \models^{\kappa} B$, $B \models^{\kappa'} C$ *and for all* $B_i, B_j \in \delta^{\kappa \cup \kappa'}(B)$ *it holds that* $A \wedge B \models^{\kappa \cup \kappa'} B_i \leftrightarrow B_j$, *then* $A \models^{\kappa \cup \kappa'} C$.

2.2 The language of UL

Subsequently, the project of a general language of underspecification will be carried out in more detail.

The *signature* of **UL** consists of the following disjoint sets

1. a set of operators $\exists$, $\forall$, $\rightarrow$, $\sharp$ (disambiguation), $\Box_1$, $\Box_2$, ... (indices)
2. a set of (first order) terms $\mathtt{t}_1$, $\mathtt{t}_2$, ..., which include a set of variables x_1, x_2, ...
3. a set of predicate symbols $\bot$ (false), P_1, P_2, ...
4. and a set of labels l_1, l_2, ...

The *syntax* of *underspecified formulas* in **UL** is defined subsequently. Note that we distinguish between *underspecified formulas* and *underspecification forms*. A basic underspecified formula is an underspecification form which is labeled with a contextual *index*. In this way, one can make sure that underspecification comes always together with a contextual parameter, which could serve to disambiguate it. Of course, the coindexing of underspecification forms makes only sense, if there is a reasonable amount of 'similarity' among the coindexed material, e.g. sharing of labels.

Atomic formula:

If $\mathtt{t}_1, \ldots, \mathtt{t}_n$ are terms and P is a predicate symbol which requires n arguments then $P(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ is an *atomic formula*.

Partial formula:

If u is an underspecified formula, and x is a variable then

1. $l_1 : \neg l_2$
2. $l_1 : u \rightarrow l_2$
3. $l_1 : \exists x.u(x) \wedge l_2$
4. $l_1 : \forall x.u(x) \rightarrow l_2$

are *partial formulas*. For a partial formula, l_1 is called the *label* of the partial formula, and l_2 is its *embedded label*.

Set B of building blocks:

B consists of labeled underspecified and partial formulas.

Basic underspecification form:

If B is a set of buildings blocks, C is a set of relational constraints then $\langle B, C\rangle$ is a *basic underspecification form*.

Complex underspecification form:

If v and v' are underspecification forms, then $v \sharp v'$ is a *complex underspecification form*.

Basic underspecified formula:

If v is an underspecification form and $\Box_i$ is an index then

$\Box_i(v)$ is a *basic underspecified formula*.

Complex underspecified formula:

If u_i is an atomic formula, or a basic or complex underspecified formula then

1. $\neg u$
2. $u_1 \wedge u_2$
3. $u_1 \vee u_2$
4. $u_1 \rightarrow u_2$
5. $\exists x.u$
6. $\forall x.u$

are *complex underspecified formulas*.

Labeled underspecified formula:

If u is an underspecified formula then $l : u$ is a *labeled underspecified formula*.

Concerning the *relational constraints*, note that the partial formulas themselves induce structural constraints, as well. E.g. the constraint $l_2 \leq l_1$ could have been derived from the partial formula $(l_1 : \forall x.u(x) \rightarrow l_2)$.

We call the set C in an underspecification form $\langle B, C\rangle$, the *explicit constraints*, and the constraints which are derived from the elements of B the *implicit constraints*. The *constraints of* $\langle B, C\rangle$, i.e. union of the explicit and the implicit relational constraints of an underspecification form $\langle B, C\rangle$, must satisfy at least the following conditions

1. The constraints must form an asymmetric relation.
2. They must ensure that all variables are bound by some quantifier. I.e. if a variable occurs in some formula, this formula must be required to be subordinate to the partial formula which contains the corresponding quantifier.
3. The minimal elements must be underspecified formulas (not partial ones).

A constraint set is *total* iff it is a total order.

Furthermore, we demand that all the constraint sets of the basic underspecification forms in a complex underspecification form $v \sharp v'$ must be mutually incompatible.

The basis of the *disambiguation device* D is given by a disambiguation function δ which maps a basic underspecification form $v := \langle B \cup \{l : p\}, C\rangle$ onto an (underspecified) formula

$$p\,[l'/\langle B, C\,[l\,/l']\rangle] \quad \text{where } l' \text{ embedded label of } p$$

if the set of constraints $\{l_i \leq l \mid l_i \text{ occurs in } v\}$ is consistent with the constraints of v. The disambiguation function δ must be total, i.e. it must provide for a value for any underspecification form.

The dependency on contextual constraints will be realized as a dependency on a value assignment to the indices of an underspecified formula. Such a *disambiguation assignment* α is a function from indices $\Box_i$ onto constraint sets C. A *disambiguation function* δ^α *with associated disambiguation assignment* α is a restriction of a disambiguation function δ to those values which are compatible with a given disambiguation assignment α, i.e.

$$\delta^\alpha(\Box_i(\langle B, C\rangle)) := \begin{cases} \delta(\langle B, C\rangle) & \text{if } \alpha(\Box_i) \text{ is compatible with the} \\ & \quad \text{(recursively embedded) constraints of } \delta(\langle B, C\rangle) \\ \bot & \text{otherwise} \end{cases} \tag{32}$$

A disambiguation assignment α' is called a *refinement* of another disambiguation assignment α, if for all indices $\Box_i$ $\alpha'(\Box_i) \models \alpha(\Box_i)$. A disambiguation assignment is called *total* if all its values are total.

Disambiguation functions δ^α with total assignments α can also disambiguate complex underspecification forms:

$$\delta^\alpha(\Box_i(v_1 \sharp v_2)) := \begin{cases} \delta^\alpha(\Box_i(v_1)) & \text{if } \delta^\alpha(\Box_i(v_1)) \neq \bot \\ \delta^\alpha(\Box_i(v_2)) & \text{if } \delta^\alpha(\Box_i(v_2)) \neq \bot \\ \bot & \text{otherwise} \end{cases} \tag{33}$$

Whenever no explicit reference to the labels is required, labels can be omitted:

long form	short form
$l_1 : \neg l_2$	$\neg$
$l_1 : u \rightarrow l_2$	$u \rightarrow$
$l_1 : \exists x.u(x) \wedge l_2$	$\exists x.u(x)$

For example, let us assume that $\forall x_1.\mathtt{boy}(x_1)$, $\exists x_2.\mathtt{movie}(x_2)$, and $\mathtt{see}(x_1, x_2)$[9] denote the meanings of *every boy*, *a movie*, and *saw*, respectively, in sentence (1). This means that an NP-meaning consists of the specification of its quantifier and its restrictor. We do not make any further stipulations which would be specific to any individual semantic theory.

2.3 The logic of UL

In this section, the notion of satisfiability of underspecified formulas will be defined and the rule of Generalized Modus Ponens will be introduced.

Definition 1 (Satisfiability of UL formulas)
Let $\mathcal{M}$ be a first order model in the usual manner with interpretations $P_\mathcal{M}$ and $\mathtt{t}_\mathcal{M}$ for predicate symbols P and for terms $\mathtt{t}$ respectively. A formula u is satisfiable in a first order model $\mathcal{M}$ if for all disambiguation functions δ^α there exists a total refinement α' of α such that one of the following cases applies

1. $\mathcal{M}, \delta^{\alpha'} \models P(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ if $\langle \mathtt{t}_{1\mathcal{M}}, \ldots, \mathtt{t}_{n\mathcal{M}}\rangle \in P_\mathcal{M}$.
   ($\mathtt{t}_1, \ldots, \mathtt{t}_n$ ground terms)

2. $\mathcal{M}, \delta^{\alpha'} \models \Box_i(\langle B, C\rangle)$ if $\mathcal{M}, \delta^{\alpha'} \models \delta^{\alpha'}(\Box_i(\langle B, C\rangle))$

3. $\mathcal{M}, \delta^{\alpha'} \models \Box_i(v \sharp v')$ if $\mathcal{M}, \delta^{\alpha'} \models \delta^{\alpha'}(\Box_i(v \sharp v'))$

4. (a) $\mathcal{M}, \delta^{\alpha'} \models \neg u$ if $\mathcal{M}, \delta^{\alpha'} \not\models u$
   (b) $\mathcal{M}, \delta^{\alpha'} \models u_1 \wedge u_2$ if $\mathcal{M}, \delta^{\alpha'} \models u_1$ and $\mathcal{M}, \delta^{\alpha'} \models u_2$

[9] We ignore tense in this paper.

(c) $\mathcal{M}, \delta^{\alpha'} \models u_1 \vee u_2$ if $\mathcal{M}, \delta^{\alpha'} \models u_1$ or $\mathcal{M}, \delta^{\alpha'} \models u_2$

(d) $\mathcal{M}, \delta^{\alpha'} \models u_1 \rightarrow u_2$ if $\mathcal{M}, \delta^{\alpha'} \not\models u_1$ or $\mathcal{M}, \delta^{\alpha'} \models u_2$

(e) $\mathcal{M}, \delta^{\alpha'} \models \exists x.u$ if for some ground term $\mathtt{t}$ $\mathcal{M}, \delta^{\alpha'} \models u\,[x/\mathtt{t}]$

(f) $\mathcal{M}, \delta^{\alpha'} \models \forall x.u$ if for every ground term $\mathtt{t}$ $\mathcal{M}, \delta^{\alpha'} \models u\,[x/\mathtt{t}]$

Note that $\sharp$ is unlike disjunction because it respects the contextual constraints given by the indices $\Box_i$. Furthermore, the condition that the constraint sets which occur in a complex underspecification form $v \sharp v'$ be mutually incompatible, guarantees for the distributivity (18) of the $\sharp$-operator since eventually at most one 'reading' will be chosen during the evaluation against a total disambiguation assignment. For example, let v be the reading which is picked by the (total) disambiguation assignment α, then

$$\begin{array}{l}\mathcal{M}, \delta^{\alpha} \models \neg\Box_i(v \sharp v') \text{ iff } \mathcal{M}, \delta^{\alpha} \not\models \Box_i(v) \sharp \Box_i(v') \\ \text{iff } \mathcal{M}, \delta^{\alpha} \not\models \delta^{\alpha}(\Box_i(v) \sharp \Box_i(v')) \text{ iff } \mathcal{M}, \delta^{\alpha} \not\models \Box_i(v) \text{ iff } \mathcal{M}, \delta^{\alpha} \models \neg\Box_i(v) \\ \text{iff } \mathcal{M}, \delta^{\alpha} \models \langle \neg\Box_i(v), \{\}\rangle \text{ iff } \mathcal{M}, \delta^{\alpha} \models \delta^{\alpha}(\langle \neg\Box_i(v), \{\}\rangle \sharp \langle \neg\Box_i(v'), \{\}\rangle) \\ \text{iff } \mathcal{M}, \delta^{\alpha} \models \langle \neg\Box_i(v), \{\}\rangle \sharp \langle \neg\Box_i(v'), \{\}\rangle\end{array} \tag{34}$$

In order to formulate side conditions on our deduction rules, we need the notion of *polarity* of a subformula, following [14]. The polarity of a formula is defined if its relative scope to all negations or monotone decreasing quantifiers in the whole formula is fixed: The formula has positive (resp. negative) polarity if it is in the scope of an even (resp. odd) number monotone decreasing quantifiers. Otherwise, its polarity is undefined.

Based on the above semantics, the following inference rule is sound, if the partial formula $\forall x.P(x)$ is restricted to have positive polarity.

$$\frac{\begin{array}{c}\Box_i(\langle\{\forall x.P(x)\} \cup B, C\rangle) \\ P(\mathtt{t})\end{array}}{\Box_i(\langle B, C\rangle)\,[x/\mathtt{t}]} \tag{35}$$

Theorem 2 *If $\forall x.P(x)$ has positive polarity, the rule (35) is sound.*

Proof. We have to show that

$$\begin{array}{c}\text{If } \mathcal{M}, \delta^{\alpha} \models \Box_i(\langle\{\forall x.P(x)\} \cup B, C\rangle) \text{ and } \mathcal{M}, \delta^{\alpha} \models P(\mathtt{t}) \\ \text{then } \mathcal{M}, \delta^{\alpha} \models \Box_i(\langle B, C\rangle)\,[x/\mathtt{t}]\end{array} \tag{36}$$

By assuming $\mathcal{M}, \delta^{\alpha} \models P(\mathtt{t})$ and applying the semantic definitions, we get

$$\begin{array}{c}\forall\ \delta^{\alpha} \text{ it must hold that} \\ \text{if } \mathcal{M}, \delta^{\alpha} \models \delta^{\alpha}(\Box_i(\langle\{\forall x.P(x)\} \cup B, C\rangle)) \\ \text{then } \mathcal{M}, \delta^{\alpha} \models \delta^{\alpha}(\Box_i\langle B, C\rangle)\,[x/\mathtt{t}]\end{array} \tag{37}$$

We show (37) by an induction on all possible disambiguated formulas (cf. [13, p.176])

$$\delta^{\alpha}(\Box_i(\langle\{\forall x.P(x)\} \cup B, C\rangle))$$

which corresponds simultaneously to an induction on

$$\delta^{\alpha}(\Box_i\langle B, C\rangle)\,[x/\mathtt{t}]$$

due to the coindexation of both formulas. The induction on possible disambiguated formulas amounts to an induction on the number e of partial formulas 'above' the formula $\forall x.P(x)$ in a disambiguated formula (i.e. the 'embedding level' of the partial formula).

$e = 0$**:** clear, since this is the usual format of the generalized modus ponens.

$e = j-1$**:** Assume that

$$\begin{array}{l}\text{if } \mathcal{M}, \delta^\alpha \models \delta^\alpha(\Box_i(\left\langle \begin{array}{l}\{l_j : \forall x.P(x) \to l_{j2}\} \cup B, \\ \{l_j \leq l_{1,\ldots,j-1}\} \cup C\end{array} \right\rangle)) \\ \text{then } \mathcal{M}, \delta^\alpha \models \delta^\alpha(\Box_i\langle B, \{l_j \leq l_{1,\ldots,j-1}\} \cup C\rangle)\,[x/\mathtt{t}]\end{array} \tag{38}$$

$e = j$**:** We have to show that

$$\begin{array}{l}\text{if } \mathcal{M}, \delta^\alpha \models \delta^\alpha(\Box_{i'}(\left\langle \begin{array}{l}\{l_j : \forall x.P(x) \to l_{j2}\} \cup B, \\ \{l_j \leq l_{1,\ldots,j-1,j+1}\} \cup C\end{array} \right\rangle)) \\ \text{then } \mathcal{M}, \delta^\alpha \models \delta^\alpha(\Box_{i'}\langle B, \{l_j \leq l_{1,\ldots,j-1,j+1}\} \cup C\rangle)\,[x/\mathtt{t}]\end{array} \tag{39}$$

Since there is no restriction on B in (38), a formula which is embedded under l_j can be removed without harm:

$$\begin{array}{l}\text{if } \mathcal{M}, \delta^\alpha \models \delta^\alpha(\Box_i(\left\langle \begin{array}{c}\{l_j : \forall x_1.P_1(x_1) \to l_{j2}\} \cup B \\ \setminus\{l_k : \exists x_2.P_2(x_2) \wedge l_{k2}\} \\ \{l_j \leq l_{1,\ldots,j-1}, l_k \leq l_j\} \cup C\end{array}, \right\rangle)) \\ \text{then } \mathcal{M}, \delta^\alpha \models \delta^\alpha(\Box_i\left\langle \begin{array}{l} B \setminus l_k : \exists x_2.P_2(x_2) \wedge l_{k2}, \\ \{l_j \leq l_{1,\ldots,j-1}, l_k \leq l_j\} \cup C\end{array} \right\rangle)\,[x_1/\mathtt{t}]\end{array} \tag{40}$$

However, then

$$\begin{array}{l}\text{if } \mathcal{M}, \delta^\alpha \models \exists x_2 \left(\begin{array}{l} P_2(x_2) \\ \wedge\, \delta^\alpha(\Box_i(\left\langle \begin{array}{c}\{l_j : \forall x_1.P_1(x_1) \to l_{j2}\} \cup B \\ \setminus\{l_k : \exists x_2.P_2(x_2) \wedge l_{k2}\} \\ \{l_j \leq l_{1,\ldots,j-1}, l_k \leq l_j\} \cup C\end{array}, \right\rangle)) \end{array} \right) \\ \quad \text{then} \\ \mathcal{M}, \delta^\alpha \models \exists x_2 \left(\begin{array}{l} P_2(x_2) \\ \wedge\, \delta^\alpha(\Box_i\left\langle \begin{array}{l} B \setminus l_k : \exists x_2.P_2(x_2) \wedge l_{k2}, \\ \{l_j \leq l_{1,\ldots,j-1}, l_k \leq l_j\} \cup C\end{array} \right\rangle) \end{array} \right)\,[x_1/\mathtt{t}]\end{array} \tag{41}$$

Similarly for other, monotone increasing partial formulas. □

Consider again (1). It can be understood as saying that each of the boys saw a different movie or that all of them saw the same movie. If in addition to (1) we know that Kevin is a boy, then we should be able to derive the fact that Kevin saw a movie – irrespective of what reading the first sentence is intended to have. The fact that Kevin is a boy is represented by $\mathtt{boy}(\mathtt{kevin})$. An application of generalized modus ponens (GMP) looks as follows.

$$\frac{\begin{array}{c}\Box_i(\langle\{\forall x_1.\mathtt{boy}(x_1), \exists x_2.\mathtt{movie}(x_2), \mathtt{see}(x_1, x_2)\}, C\rangle) \\ \mathtt{boy}(\mathtt{kevin})\end{array}}{\Box_i(\langle\{\exists x_2.\mathtt{movie}(x_2), \mathtt{see}(\mathtt{kevin}, x_2)\}, C\rangle)} \tag{42}$$

As the conclusion is not ambiguous any more there is no problem with correctness here. But assume we had an ambiguous relative clause ϕ attached to *movie* (e.g., *in which he didn't like every actor*) then we need to guarantee correctness by contextual restrictions. Modus Ponens must, therefore, mark the consequent and the clause it derives from as correlated.

Inferences in the context of negation will be discussed later, in the section 4.

3 Instantiations of UL

In order to be more specific, let's look at LFG semantics as a particular instantiation of our abstract logical language[10]. Since the LFG semantics does not provide for a coindexation mechanism, we add the indices $\Box_i$ of **UL** to it.

[10]Other instances are discussed in the full version of this paper [10].

3.1 LFG

In [6]'s linear logic approach, modus ponens manipulates meaning constructors:

$$\frac{\begin{array}{c}\Box_i(\{\forall \mathrm{G}, S.(\forall x.g_\sigma \rightsquigarrow x \multimap \mathrm{G} \rightsquigarrow S(x)) \multimap \mathrm{G} \rightsquigarrow \texttt{every}(P,S)\} \cup \phi) \\ P(\texttt{t})\end{array}}{\Box_i(\{g_\sigma \rightsquigarrow \texttt{t}\} \cup \phi)} \tag{43}$$

For example, if we instantiate the first premiss with the meaning constructor for *every boy* and the second premiss with `boy(kevin)`, we get that the subject of the derived sentence is Kevin, i.e. $g_\sigma \rightsquigarrow \texttt{kevin}$.

Soundness can be proved by mapping the LFG linear logic formulas to **UL** formulas[11] via a translation function τ_{LL}, where $\langle Q, Op\rangle \in \{\langle \exists, \wedge\rangle, \langle \forall, \rightarrow\rangle\}$:

$$\begin{array}{l}\tau_{LL}(\forall \mathrm{G}, S.(\forall x.g_\sigma \rightsquigarrow x \multimap \mathrm{G} \rightsquigarrow S(x)) \multimap \mathrm{G} \rightsquigarrow Q(R,S)) \\ \quad := l_g : Qx.R(x)\ Op\ l_S\end{array} \tag{44}$$

We may go even further and formulate deduction rules that operate on f-structures themselves, as suggested by the map τ_{FS}:

$$\tau_{FS}\left(\left[\ \mathrm{G}\ g : \left[\begin{array}{l}\mathrm{SPEC}\ Q \\ \mathrm{PRED}\ R\end{array}\right]\ \right]\right) := l_g : Qx.R(x)\ Op\ l_S \tag{45}$$

An application of modus ponens to (2) would then result in the instantiation of

$$\left[\ \mathrm{SUBJ}\ g : \left[\begin{array}{l}\mathrm{SPEC}\ \texttt{every} \\ \mathrm{PRED}\ \texttt{boy}\end{array}\right]\ \right] \tag{46}$$

to $[\ \mathrm{PRED}\ \texttt{kevin}\ ]$ yielding

$$f : \left[\begin{array}{l}\mathrm{PRED}\ \texttt{see} \\ \mathrm{SUBJ}\ \ g : [\ \mathrm{PRED}\ \texttt{kevin}\ ] \\ \mathrm{OBJ}\ \ \ h : \left[\begin{array}{l}\mathrm{SPEC}\ \texttt{a} \\ \mathrm{PRED}\ \texttt{movie}\end{array}\right]\end{array}\right] \tag{47}$$

4 Negation

As mentioned earlier, an application of GMP (Generalized Modus Ponens) is only correct in the absence of negations, or monotone decreasing quantifiers.

$$\frac{\begin{array}{c}\textit{John is a politician.} \\ \textit{At least one problem preoccupies every politician.}\end{array}}{\textit{At least one problem preoccupies John.}} \tag{48}$$

$$\begin{array}{c}\textit{John is a politician.} \\ \textit{Few problems preoccupy every politician.} \\ \not\forall\ \textit{Few problems preoccupy John.}\end{array} \tag{49}$$

$$\begin{array}{c}\textit{John is a politician.} \\ \textit{Every politician doesn't sleep.} \\ \not\forall\ \textit{John doesn't sleep.}\end{array} \tag{50}$$

The examples in (48), (49), and (50) show that GMP may only be applied to an element γ of B, if there is no disambiguation of $\langle B, C\rangle$ (by D) that assigns γ narrow scope with respect to (an odd number of occurrences of) negations and monotone decreasing quantifiers, i.e. to cases where γ has positive polarity.

[11] For an in-depth investigation of mappings among underspecified semantic representation formalisms, see the contribution by Crouch and van Genabith in [3].

Thus the occurrence $\forall x.\mathtt{boy}(x)$ in (1) has positive polarity, whereas it has negative polarity in (51) and indefinite polarity in (52).

Few mothers believed that every boy saw a movie. (51)

Every boy didn't see a movie. (52)

Subsequently, polarities, $+, -$, or i(ndefinite), will be superscripted to the labels of the partial formulas. Now, let us concentrate on (52). We may split its completely underspecified representation

$$\langle \left\{ \begin{array}{c} l_{1^i} : \forall x_1.\mathtt{boy}(x_1) \rightarrow l_{12}, \\ l_{2^i} : \exists x_2.\mathtt{movie}(x_2) \wedge l_{22}, \\ l_{3^+} : \neg l_{31}, \\ l_4 : \mathtt{see}(x_1, x_2) \end{array} \right\}, \{l_4 \leq l_{12,22,31}\} \rangle \tag{53}$$

into two underspecified representations in which $\forall x.\mathtt{boy}(x)$ has definite polarity ($\{l_4 \leq l_{12,22,31}\}$ abbreviates the set $\{l_4 \leq l_{12}, l_4 \leq l_{22}, l_4 \leq l_{31}\}$.)

$$\langle \left\{ \begin{array}{c} l_{1^+} : \forall x_1.\mathtt{boy}(x_1) \rightarrow l_{12}, \\ l_{2^i} : \exists x_2.\mathtt{movie}(x_2) \wedge l_{22}, \\ l_{3^+} : \neg l_{31}, \\ l_4 : \mathtt{see}(x_1, x_2) \end{array} \right\}, \{l_3 \leq l_{12}, l_4 \leq l_{12,22,31}\} \rangle \tag{54}$$

$$\langle \left\{ \begin{array}{c} l_{1^-} : \forall x_1.\mathtt{boy}(x_1) \rightarrow l_{12}, \\ l_{2^i} : \exists x_2.\mathtt{movie}(x_2) \wedge l_{22}, \\ l_{3^+} : \neg l_{31}, \\ l_4 : \mathtt{see}(x_1, x_2) \end{array} \right\}, \{l_1 \leq l_{31}, l_4 \leq l_{12,22,31}\} \rangle \tag{55}$$

(54) represents the set of meanings of (52) in which the negation has narrow scope with respect to the universal quantifier, and (55) the set of meanings in which it has wide scope. These two descriptions can be combined into one formula with the help of the connective $\sharp$. The introduction rule (and simultaneously the elimination rule) for the $\sharp$-operator in (56) states that an underspecified formula may be replaced by two $\sharp$-connected underspecified formulas for the same set B of building blocks with orthogonal constraint sets $C \cup C_1$ and $C \cup C_2$ which together describe exactly the set of readings admitted by C.

$$\frac{\Box_i(\langle B, C \rangle)}{\Box_i(\langle B, C \cup C_1 \rangle) \sharp \Box_i(\langle B, C \cup C_2 \rangle)} \text{ if } \models C \cup C_1 \text{ iff } \not\models C \cup C_2 \tag{56}$$

It is the $\sharp$-operator which makes the formalism *cu-deductive*. Disambiguation steps and inference steps can alternate, while always producing meaningful formulas. On this basis (53) can be rewritten equivalently by (57) as a combination of an underspecified formula where the negation has narrow scope wrt. the universally quantified NP in the left subformula and wide scope in the right subformula.

$$\begin{array}{l} \langle \left\{ \begin{array}{c} l_{1^+} : \forall x_1.\mathtt{boy}(x_1) \rightarrow l_{12}, \\ l_{2^i} : \exists x_2.\mathtt{movie}(x_2) \wedge l_{22}, \\ l_{3^+} : \neg l_{31}, \\ l_4 : \mathtt{see}(x_1, x_2) \end{array} \right\}, \{l_3 \leq l_{12}, l_4 \leq l_{12,22,31}\} \rangle \\ \sharp \langle \left\{ \begin{array}{c} l_{1^-} : \forall x_1.\mathtt{boy}(x_1) \rightarrow l_{12}, \\ l_{2^i} : \exists x_2.\mathtt{movie}(x_2) \wedge l_{22}, \\ l_{3^+} : \neg l_{31}, \\ l_4 : \mathtt{see}(x_1, x_2) \end{array} \right\}, \{l_1 \leq l_{31}, l_4 \leq l_{12,22,31}\} \rangle \end{array} \tag{57}$$

What can we infer from (57) resp. (53)? By the usual equivalence transformations, the universal quantifier with negative polarity can be replaced by an existential one with positive

polarity.

$$\begin{array}{l}\langle\left\{\begin{array}{c} l_{1^+} : \forall x_1.\mathtt{boy}(x_1) \rightarrow l_{12}, \\ l_{2^i} : \exists x_2.\mathtt{movie}(x_2) \wedge l_{22}, \\ l_{3^+} : \neg l_{31}, \\ l_4 : \mathtt{see}(x_1, x_2) \end{array}\right\}, \{l_3 \leq l_{12}, l_4 \leq l_{12,22,31}\}\rangle \\ \sharp\langle\left\{\begin{array}{c} l_{1^+} : \exists x_1.\mathtt{boy}(x_1) \wedge l_{12}, \\ l_{2^i} : \exists x_2.\mathtt{movie}(x_2) \wedge l_{22}, \\ l_{3^+} : \neg l_{31}, \\ l_4 : \mathtt{see}(x_1, x_2) \end{array}\right\}, \{l_3 \leq l_{12}, l_4 \leq l_{12,22,31}\}\rangle \end{array} \tag{58}$$

We can weaken $l_{1^+} : \exists x_1.\mathtt{boy}(x_1) \wedge l_{12}$ to $l_{1^+} : \exists x_1.\mathtt{boy}(x_1) \rightarrow l_{12}$ and realize now that the right subformula of $\sharp$ follows from the left one. Hence, we conclude

$$\langle\left\{\begin{array}{c} l_{1^+} : \exists x_1.\mathtt{boy}(x_1) \rightarrow l_{12}, \\ l_{2^i} : \exists x_2.\mathtt{movie}(x_2) \wedge l_{22}, \\ l_{3^+} : \neg l_{31}, \\ l_4 : \mathtt{see}(x_1, x_2) \end{array}\right\}, \{l_3 \leq l_{12}, l_4 \leq l_{12,22,31}\}\rangle \tag{59}$$

This means

There exists an x with the property that
if x is a boy then x didn't see a movie (60)

is a consequence of (52).

5 Possible extensions

For the sake of simplicity we have restricted ourselves to scope ambiguities. We have to admit coindexed atomic formulas in order to represent correlated lexical meanings. For example, one might want to express that a lexically ambiguous word like *plant* means *factory* all over a text, instead of a designating living things like flowers in some occurrences and factories in other occurrences. Hence, the syntax of **UL** will be extended by one more kind of basic underspecified formulas:

$$\Box_k(P(\mathtt{t}_1, \ldots, \mathtt{t}_n)) \tag{61}$$

The satisfiability condition will be analogous to the existing ones for basic underspecified formulas.

In order to account for plural ambiguities like the one in (12), one needs to extend the disambiguation device by rules which introduce new formula material when a plural is disambiguated to a collective reading or to a distributive reading etc. Similarly, the disambiguation device has to consider interaction postulates which allow the meaning representation of an indefinite NP in an embedded underspecified formula, e.g. in a relative clause, to get wide scope over parts of the meaning representation of the matrix clause. Consider the example taken from [13]

Every student to whom every professor recommends
a certain book which the student has already read is lucky. (62)

6 Conclusion

We presented a general inference scheme for an underspecified semantic representation formalism **UL** and showed how this inference scheme can be specialized to the underspecified representation languages of linear logic formulas and f-structures in LFG semantics. The benefit is that the logical properties of the inference scheme can be investigated on an abstract level, and are then 'inherited' by those formalisms which fulfill the necessary requirements. For example, the novel property of the **UL** formalism of being cu-deductive, i.e. of allowing for the alternation of disambiguation steps and proper inference steps, could be ported to those underspecified semantic representation formalisms which come with a well-defined disambiguation method.

Since we assume a very general language for expressing the structural constraints in an underspecified representation, there is enough room for extensions. For example, the structural constraints could mirror as well syntactic conditions as well as semantic requirements on the set of fully disambiguated representations.

References

[1] Hiyan Alshawi, editor. *The Core Language Engine.* ACL-MIT Press Series in Natural Languages Processing. MIT Press, Cambridge, Mass., 1992.

[2] Hiyan Alshawi and Richard Crouch. Monotonic semantic interpretation. In *Proceedings of ACL*, pages 32–39, Newark, Delaware, 1992.

[3] Robin Cooper, Dick Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspars, Hans Kamp, Manfred Pinkal, Massimo Poesio, and Steve Pulman. Building the framework. Deliverable 15, FraCaS, LRE 62-051, University of Edinburgh, 1996.

[4] Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. Translation using minimal recursion semantics. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, University of Leuven, Belgium, 1995.

[5] Richard Crouch. Ellipsis and quantification: A substitutional approach. In *Proceedings of the 6th Meeting of the Association for Computational Linguistics, European Chapter*, Dublin, 1995.

[6] Mary Dalrymple, John Lamping, Fernando C.N. Pereira, and Vijay Saraswat. A deductive account of quantification in LFG. In Makoto Kanazawa, Christopher J. Pinon, and Henriette de Swart, editors, *Quantifiers, Deduction and Context.* CSLI, Stanford, Ca., 1995.

[7] David R. Dowty, Robert E. Wall, and Stanley Peters. *Introduction to Montague Semantics.* Reidel, Dordrecht, Holland, 1981.

[8] Anette Frank and Uwe Reyle. Principle based semantics for HPSG. In *Proceedings of the 6th Meeting of the Association for Computational Linguistics, European Chapter*, Dublin, 1995.

[9] Jerry Hobbs and Stuart M. Shieber. An algorithm for generating quantifier scopings. *Computational Linguistics*, 13:47–63, 1987.

[10] Esther König and Uwe Reyle. A general reasoning scheme for underspecified representations. In Hans-Jürgen Ohlbach and Uwe Reyle, editors, *Logic and its Applications. Festschrift for Dov Gabbay. Part I.* Kluwer, 1996.

[11] Fernando C.N. Pereira. Categorial semantics and scoping. *Computational Linguistics*, 16(1):1–10, 1990.

[12] Carl Pollard and Ivan A. Sag. *Head Driven Phrase Structure Grammar.* University of Chicago Press, Chicago, 1994.

[13] Uwe Reyle. Dealing with ambiguities by underspecification: Construction, representation, and deduction. *Journal of Semantics*, 10(2):123–179, 1993.

[14] Uwe Reyle. On reasoning with ambiguities. In *Proceedings of the 6th Meeting of the Association for Computational Linguistics, European Chapter*, pages 1–8, Dublin, 1995.