Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

Ambiguity and reasoning

D.J.N. Eijck and J. Jaspars

Computer Science/Department of Software Technology

CS-R9616 1996

Report CS-R9616
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Ambiguity and Reasoning

Jan van Eijck[§] and Jan Jaspars[‡]

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

[§] jve@cwi.nl [‡] jaspars@cwi.nl

Abstract

In this paper, reasoning with ambiguous representations is explored in a formal way, with ambiguities at the level of propositions in propositional logic and predicate logic, and ambiguous representations of scopings in predicate logic as the main examples. First a version of propositional logic with propositional ambiguities is presented and a sequent axiomatization for it is given. This is then extended to predicate logic. Next, predicate logic with scope ambiguities is introduced and discussed, and again a sequent calculus for it is proposed. The conclusion connects the results to natural language semantics, and briefly compares them with existing logics of ambiguity. An appendix gives completeness proofs for our versions of ambiguous propositional and predicate logic.

AMS Subject Classification (1991): 03B65, 03B80, 68S05, 68T30, 92K20.
CR Subject Classification (1991): F.3.1, F.3.2, I.2.1, I.2.4, I.2.7.
Keyword and Phrases: Semantics of Natural Language, Reasoning with Underspecified Forms, Knowledge Representation Languages.
Note: This research was sponsored by CEC-project LRE-62-01 (FraCaS).

1. Representing Ambiguity

In the formal study of natural language semantics the representation of ambiguous information is one of the major problems. Initial representations of NL expressions are often ambiguous, due to lack of information about the meanings of lexical items (lexical ambiguity), the ways in which anaphoric elements are to be resolved (anaphoric under-specification), attachment ambiguities (structural ambiguity) and the choice between various possible scope orderings between operators (scope ambiguity).

Current ambiguous representation formalisms have evolved from highly expressive representation languages, which usually do not come equipped with a deductive calculus (so strictly speaking they are not 'logics'). It seems hopeless to start the investigation of reasoning with ambiguity from this end. Instead, we start from the other end, by extending well understood languages like the languages of propositional and predicate logic with an ambiguation operator. It turns out to be fairly easy to axiomatize such logics and to develop an account of reasoning with ambiguity.

Let us look at how some current representation languages deal with the ambiguity of example sentence (1).

1 *Every boy did not appear.*

The representation language of the Core Language Engine described in [2], the language of Quasi Logical Forms or QLFs, deals with under-specification of quantifier scope by representing quantifiers as 'quasi-terms' in argument position. The QLF representation of (1) is given in (2).

2 $\neg A(qterm, \forall, x, Bx)$.

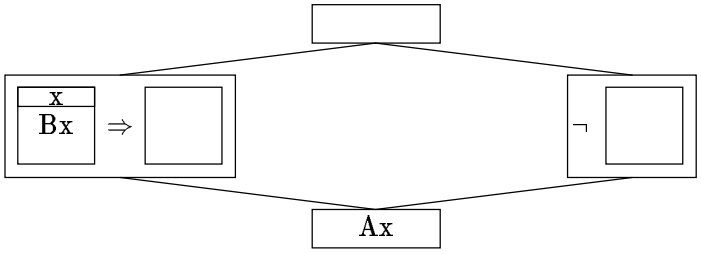

Figure 1: Under-specified DRS for 'Every boy did not appear.'

The under-specified discourse representation of (1) (see [19] for an account of UDRT) is given in Figure 1. Under-specified DRT is the result of under-specifying the relation of subordination for DRS boxes. In the example figure, the vertical lines connecting the boxes indicate subordination. The subordination relation between box-implication and box-negation is left under-specified.

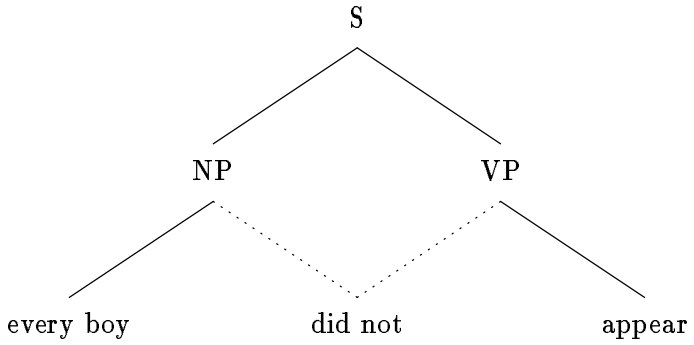

Figure 2: Under-specified syntax tree for 'Every boy did not appear' in Flexible MG.

In flexible Montague Grammar [10] the example sentence (1) is structurally ambiguous: it has a tree where *did not* is part of the VP and one in which *did not* is part of the NP. We can take these two trees together in one partially under-specified tree, as in Figure 2. The dotted lines in the tree indicate that the domination relations in the syntactic tree are not fully determined. The leaf node with the auxiliary 'did not' has not made up its mind about which node in the tree it considers as its mother. If the auxiliary is adopted by the NP, this

forces the semantic type of 'did not' to be $(((e,t),t),((e,t),t))$ (a map from NP denotations to NP denotations), with corresponding translation $\lambda\mathbf{P}\lambda Q\neg\mathbf{P}Q$. If the auxiliary is adopted by the VP, its semantic type is forced to be $((e,t),(e,t))$ (a map from VP denotations to VP denotations), with corresponding translation $\lambda P\lambda x\neg Px$.

The essence of ambiguous operator representation seems to be that the order of application of operators can remain under-specified. QLF assumes that the operators are always quantifiers, which is obviously too limited. UDRT defines under-specified representations by means of a partial specification of the subordination relation between boxes. This formulation turns out to engender serious complications in the design of a proof system. Also, UDRT suggests a strong connection between the dynamism of the DRT treatment of reference markers and the treatment of operator ambiguity (a suggestion which is taken over in Muskens [16]). As we will show below, operator ambiguity and dynamic treatment of variables are orthogonal issues: it is very well possible to combine under-specified operator orderings with a static treatment of variables.

We will build our logics of ambiguity on top of ordinary propositional and predicate logic. In line with Alshawi and Crouch's monotonic semantics for quasi-logical form, we allow the mapping of natural language expressions to logical form to be a relation—a function to sets of logical forms—rather than an ordinary function. This reflects the ambiguity of the original expressions. We will take as our motto 'try to understand simple things first.' The main outcome of our investigation will be that any logical language can be ambiguated in this simple and systematic way, and that the ambiguated language is conservative over the original language in the sense that no new conclusions in the old language can be drawn by employing the reasoning system for the enriched language. A calculus for the original language can be extended to one for the ambiguous language by adding a set of axioms or rules to deal with the ambiguous operators. Before we embark on the analysis of scope ambiguity, there are some preliminary questions to be asked about the meaning of ambiguous questions and about the analysis of local propositional ambiguities.

2. What Do Ambiguous Expressions Mean?

The principal reason for wanting to construct a meaning representation for a natural language sentence is to get a handle on the information conveyed by that sentence. Is the sentence consistent with a given body of information? If the sentence is true, what does follow from it? If a natural language sentence is ambiguous, as many natural language sentences are, the key question becomes: how can we find a representation for it that we can reason with?

There are many kinds of ambiguity in natural language. The most 'local' ambiguities are lexical ambiguities, like the one in *The ball was splendid* or *I went to the bank*, and referential ambiguities, like *John addressed her*, when there is a fixed list of possible antecedents for the pronoun. A different kind of ambiguity has to do with scope under-specification caused by the interaction of parts of speech. Examples are *Every boy didn't appear* or *Everybody in this room has to sign one document.* Related are ambiguities of distribution, as in *The boys ordered two sandwiches*, where it is left unspecified whether the object distributes over the subject (two sandwiches each) or not (two sandwiches altogether). Finally, there is an open-ended spectrum of under-specification caused by some kind of incompleteness or flaw

in the linguistic data (even: corruption of the data). Under this heading we have structural ambiguities, like the two readings of *John saw the girl with the telescope*, or the problem of what to make of *John ...(noise) ...the girl with the telescope.* In this paper we will concentrate on local ambiguity and scope ambiguity.

Suppose a sentence A is ambiguous between readings A_1 and A_2. Here are some desiderata for 'what A means'.

- If someone informs us that A is true, then one should be allowed to conclude that at least one of A_1, A_2 is true.
- If one is sure that A_1 and A_2 are both true, then one can safely assert that A is true, the ambiguity of A notwithstanding.
- If someone informs us that *not* A is true, then one should be allowed to conclude that at least one of *not* A_1, *not* A_2 is true.
- If one is sure that neither A_1 nor A_2 is true, then one can safely assert that *not* A, the ambiguity of A notwithstanding.
- Unless A_1 and A_2 are logically equivalent, A *or not* A cannot be a logical truth.
- Unless A_1 and A_2 are logically equivalent, A *and not* A need not be a contradiction.

To explain this final point a bit further, note that in this example we do not insist that several occurrences of the same expressions be disambiguated in the same way. To take a real-life example, consider 3.

3 *Every boy did not appear, and it is not the case that every boy did not appear.*

If both occurrences of the ambiguous *Every boy did not appear* in example (3) are disambiguated in the same way, then this example sentence is indeed contradictory. If we do not insist on this, however, then it is not.

Our discussion yields the following wish list for a relation of ambiguous consequence $\models_a$:

$$\begin{array}{rcl} A & \models_a & A_1, A_2 \\ A_1, A_2 & \models_a & A \\ \neg A & \models_a & \neg A_1, \neg A_2 \\ \neg A_1, \neg A_2 & \models_a & \neg A \\ A, \neg A & \not\models_a & \bot \\ \top & \not\models_a & A, \neg A \end{array}$$

Here the premisses should be read conjunctively, and the conclusions disjunctively. Note that treating ambiguity as object level disjunction will not yield the above results, for if A is made equivalent to $A_1 \vee A_2$, then this entails that $\neg A$ will be equivalent between $\neg A_1 \wedge \neg A_2$, which gives the wrong interplay between ambiguity and wide scope negation (compare the argument against ambiguity as object level disjunction in Van Deemter [6]).

Interpreting ambiguous expressions in a model M can give rise to three kinds of situations:

1. Every reading of A is true in M. Notation: $M \models A$.
2. Every reading of A is false in M. Notation: $M \dashv A$.
3. At least one reading of A is false in M, and at least one reading of A is true in M. Notation: $M \not\models A$ and $M \not\dashv A$.

Define a relation $\Gamma \models_a \Delta$ (Γ and Δ are sets of formulas) by means of:

$$\begin{array}{ll} \Gamma \models_a \Delta & :\Longleftrightarrow \quad \text{for all models } M: \\ & \text{if } M \models \gamma \text{ for all } \gamma \in \Gamma, \text{ then } M \models \delta \text{ for some } \delta \in \Delta, \\ \text{and} & \text{if } M \dashv \delta \text{ for all } \delta \in \Delta, \text{ then } M \dashv \gamma \text{ for some } \gamma \in \Gamma. \end{array}$$

It should be noted immediately that this definition of $\Gamma \models_a \Delta$ is equivalent to:

$$\begin{array}{ll} \Gamma \models_a \Delta & :\Longleftrightarrow \quad \text{for all models } M: \\ & \text{if } M \not\models \gamma \text{ for all } \gamma \in \Gamma, \text{ then } M \not\models \delta \text{ for some } \delta \in \Delta, \\ \text{and} & \text{if } M \not\dashv \delta \text{ for all } \delta \in \Delta, \text{ then } M \not\dashv \gamma \text{ for some } \gamma \in \Gamma. \end{array}$$

Also, note that the 'double barrelled' consequence relation (in fact a well-known consequence relation in partial logic [4, 15]) gives us the desired contraposition behaviour.

Immediately from the definition we have that $\Gamma \models_a \Delta$ implies that $\neg\Delta \models_a \neg\Gamma$ (where $\neg\Delta := \{\neg\delta \mid \delta \in \Delta\}$ and $\neg\Gamma := \{\neg\gamma \mid \gamma \in \Gamma\}$).

3. Propositional Logic with Local Ambiguities

Assuming that p ranges over a set of atomic propositions P, the language of ambiguous propositional logic is given by the following BNF definition:

$$\varphi \quad ::= \quad \bot \mid p \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \mathbf{?}(\varphi_1, \varphi_2)$$

Assuming that $V : P \to \{0, 1\}$ is a propositional valuation, the semantics is given by:

$$\begin{array}{rll} V \not\models \bot & & \\ V \dashv \bot & & \\ V \models p & \text{iff} & V(p) = 1 \\ V \dashv p & \text{iff} & V(p) = 0 \\ V \models \neg\varphi & \text{iff} & V \dashv \varphi \\ V \dashv \neg\varphi & \text{iff} & V \models \varphi \\ V \models (\varphi_1 \wedge \varphi_2) & \text{iff} & V \models \varphi_1 \text{ and } V \models \varphi_2 \\ V \dashv (\varphi_1 \wedge \varphi_2) & \text{iff} & V \dashv \varphi_1 \text{ or } V \dashv \varphi_2 \\ V \models \mathbf{?}(\varphi_1, \varphi_2) & \text{iff} & V \models \varphi_1 \text{ and } V \models \varphi_2 \\ V \dashv \mathbf{?}(\varphi_1, \varphi_2) & \text{iff} & V \dashv \varphi_1 \text{ and } V \dashv \varphi_2 \end{array}$$

We will use $V \models \Gamma$ as shorthand for 'for all $\gamma \in \Gamma : V \models \gamma$' , and similarly for $V \not\models \Gamma$.

It is clear from the definition that ambiguation is associative in the sense that $?(?(\varphi_1, \varphi_2), \varphi_3)$ is equivalent to $?(\varphi_1, ?(\varphi_2, \varphi_3))$. Therefore, we can agree to write both $?(?(\varphi_1, \varphi_2), \varphi_3)$ and $?(\varphi_1, ?(\varphi_2, \varphi_3))$ as $?(\varphi_1, \varphi_2, \varphi_3)$, and in general, we can write finite ambiguity lists as $?(\varphi_1, \ldots, \varphi_n)$. Also, we will write $\neg\bot$ as $\top$.

Our definition of truth and falsity as given in the table above is the same as van Deemter's notion [6] of *strong truth* and *strong falsity*, respectively. Van Deemter calls an ambiguous expression $?(\varphi_1, \varphi_2)$ *weakly* true (false) if at least one of the arguments φ_1 and φ_2 is true (false). Van Deemter discusses four different notions of entailment on the basis of strong and weak truth-values: $\models_{xy}$ with $x, y \in \{\forall, \exists\}$. The values $\forall$ and $\exists$ refer to strong truth and weak truth, respectively. For example, $\varphi \models_{\forall\exists} \psi$ says that for every valuation at which φ is strongly true ψ is weakly true. The double–barreled entailment relation that we use is (a sequent generalization of) the intersection of the relations $\models_{\forall\forall}$ and $\models_{\exists\exists}$ (note that 'not strongly true/false' is the same as 'weakly false/true').[1]

A Gentzen axiomatization

Recall the Gentzen sequent axiomatization of propositional logic given below (we write Γ, φ for $\Gamma \cup \{\varphi\}$, and so on):

Start Axiom

$$\varphi \vdash \varphi$$

Monotonicity Rules

$$\frac{\Gamma \vdash \Delta}{\Gamma, \varphi \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \varphi}$$

Cut Rule

$$\frac{\Gamma, \varphi \vdash \Delta \qquad \Gamma \vdash \Delta, \varphi}{\Gamma \vdash \Delta}$$

These rules are called the *structural rules*. Note that the fact that we take the variables Γ and Δ to range over sets rather than multi-sets or lists allows us to do without contraction rules.

The rest of the rules specify how the connectives can be introduced.

$\bot$ *Rule*

$$\Gamma, \bot \vdash \Delta$$

[1] An extensive survey of this type of entailment relation in partial logic can be found in [21].

Negation Rules

$$\frac{\Gamma \vdash \Delta, \varphi}{\Gamma, \neg\varphi \vdash \Delta}\,\neg l \qquad\qquad \frac{\Gamma, \varphi \vdash \Delta}{\Gamma \vdash \Delta, \neg\varphi}\,\neg r$$

Conjunction Rules

$$\frac{\Gamma, \varphi_1, \varphi_2 \vdash \Delta}{\Gamma, (\varphi_1 \wedge \varphi_2) \vdash \Delta}\,\wedge l \qquad\qquad \frac{\Gamma \vdash \Delta, \varphi_1 \qquad \Gamma \vdash \Delta, \varphi_2}{\Gamma \vdash \Delta, (\varphi_1 \wedge \varphi_2)}\,\wedge r$$

That is all there is to the sequent calculus for classical propositional logic. Note that we take the left and right members of the derivability relation to range over sets, so we do not have to state permutation or contraction rules (see, e.g., Smullyan [20] for discussion). We say that $\Gamma \vdash_c \Delta$ if $\Gamma \vdash \Delta$ can be derived in a finite number of steps from the start axioms by means of the sequent rules given above.

We will write $\emptyset \vdash \Delta$ as $\vdash \Delta$, and $\Gamma \vdash \emptyset$ as $\Gamma \vdash$. Recall that an empty conjunction (an empty position on the left hand side of $\vdash$) is equivalent to $\top$, and an empty disjunction (an empty position on the right hand side of $\vdash$) to $\bot$.

It is clear that the negation rules of classical propositional logic are too strong for our present purposes, for we have, for arbitrary φ:

$$\frac{\varphi \vdash \varphi}{\varphi, \neg\varphi \vdash}\,\wedge l \qquad\qquad \frac{\varphi \vdash \varphi}{\vdash \varphi, \neg\varphi}\,\wedge r$$

We now turn to the calculus for ambiguous propositional logic. We will establish soundness of the rules (in those few cases where it is not completely obvious) as we go along. Its structural rules consists of the structural rules (*start*, *monotonicity* and *cut*) of classical propositional logic and the additional rule of *contraposition*:

Contraposition Rule

$$\frac{\Gamma \vdash \Delta}{\neg\Delta \vdash \neg\Gamma}\,\text{contrap}$$

Soundness: immediate from the fact that the relation $\models_a$ is double-barrelled.[2] As may be expected, the changes for the connectives come with the treatment of negation. Both $\neg r$ and $\neg l$ are invalid. This lack of negative reasoning power is partly compensated by the rules of *double negation*:

[2]The semantic motivation to call the rule of contraposition structural is the higher independence of falsity in this setting. In the setting of partial logic so-called quadrant axiomatizations have been introduced, e.g., in [8], as a generalization of sequential axiomatizations in order to exemplify the independence of falsity. In this style of axiomatization the rule of contraposition is a truly structural rule, i.e., without occurrences of negations or other connectives.

Double Negation Rules

$$\frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \neg\neg\varphi \vdash \Delta}\,\neg\neg l \qquad\qquad \frac{\Gamma \vdash \Delta, \varphi}{\Gamma \vdash \Delta, \neg\neg\varphi}\,\neg\neg r$$

Furthermore, negations behave in a classical way as long as they have atomic arguments:

Atom Negation Rule

$$\frac{\Gamma \vdash \Delta, p}{\Gamma, \neg p \vdash \Delta}\,\neg l \text{ for atoms}$$

Soundness: immediate from the fact that for every $p \in P$, every valuation V, $V \models p$ iff not $V \models\!\!\!| p$. As we will see later, $\neg r$ for atoms can be derived from this rule together with *contraposition*. The final negation rule that we need is a combination of the classical rules $\neg l$ and $\neg r$.

Negation Swap Rule

$$\frac{\Gamma \vdash \Delta, \varphi_1 \qquad\qquad \Gamma, \varphi_2 \vdash \Delta}{\Gamma, \neg\varphi_1 \vdash \Delta, \neg\varphi_2}\,\neg\text{swap}$$

Soundness: suppose $V \models \Gamma$ and $V \models\!\!\!| \varphi$. Then we get from the soundness of the first premise (left to right direction) that $V \models \delta$ for some $\delta \in \Delta$. This takes care of the direction from left to right. To establish the other direction, use the soundness of the second premise, from right to left.

The rules for conjunction and falsum are the same as in the axiomatization of classical logic. For the ambiguation connective $?(\ldots)$ we have the following straightforward rules:

Ambiguation Rules

$$\frac{\Gamma, \varphi_1 \vdash \Delta \qquad\qquad \Gamma, \varphi_2 \vdash \Delta}{\Gamma, ?(\varphi_1, \varphi_2) \vdash \Delta}\,?l \qquad\qquad \frac{\Gamma \vdash \Delta, \varphi_1 \qquad\qquad \Gamma \vdash \Delta, \varphi_2}{\Gamma \vdash \Delta, ?(\varphi_1, \varphi_2)}\,?r$$

Soundness of left ambiguation: Suppose $V \models \Gamma$ and $V \models ?(\varphi_1, \varphi_2)$. Then $V \models \varphi_1$, and by the soundness of the first premise we have that $V \models \delta$, for some $\delta \in \Delta$. For the falsity direction, suppose $V \models\!\!\!| \Delta$. Then by the soundness of the first premise, either $V \models\!\!\!| \gamma$ for some $\gamma \in \Gamma$, or $V \models\!\!\!| \varphi_1$. By the soundness of the second premise, either $V \models\!\!\!| \gamma$ for some $\gamma \in \Gamma$, or $V \models\!\!\!| \varphi_2$. Combining this we get that either $V \models\!\!\!| \gamma$ for some $\gamma \in \Gamma$, or $V \models\!\!\!| \varphi_1$ and $V \models\!\!\!| \varphi_2$, in other words, $V \models\!\!\!| \gamma$ for some $\gamma \in \Gamma$, or $V \models\!\!\!| ?(\varphi_1, \varphi_2)$. The soundness of right ambiguation is established by an analogous piece of reasoning.

We write $\Gamma \vdash_a \Delta$ if $\Gamma \vdash \Delta$ can be derived by a finite number of applications of the structural rules with contraposition, the double negation rules $\neg\neg l$ and $\neg\neg r$, the left negation rule for atoms, the rule $\neg$swap and the ambiguation rules $?l$ and $?r$.

This completes our presentation of the calculus. We conclude the section with a few remarks about what is derivable in the proof system and what is not.

Note that we have indeed weakened negation: if φ is ambiguous, we have $\varphi, \neg\varphi \nvdash$ and $\nvdash \varphi, \neg\varphi$.

Secondly, note that the following proof sequent follows from the negation swap rule, by taking $\Gamma = \{\varphi_1\}$ and $\Delta = \{\varphi_2\}$:

$$\varphi_1, \neg\varphi_1 \vdash_a \varphi_2, \neg\varphi_2$$

The following rules are derivable:

$$\frac{\Gamma, \neg\neg\varphi \vdash \Delta}{\Gamma, \varphi \vdash \Delta}\,\neg\neg \qquad\qquad \frac{\Gamma \vdash \Delta, \neg\neg\varphi}{\Gamma \vdash \Delta, \varphi}\,\neg\neg$$

Here is the derivation for the leftmost one (the other one is derived similarly):

$$\dfrac{\dfrac{\Gamma, \neg\neg\varphi \vdash \Delta}{\Gamma, \varphi, \neg\neg\varphi \vdash \Delta}\,\text{mon} \qquad \dfrac{\dfrac{\varphi \vdash \varphi}{\varphi \vdash \neg\neg\varphi}\,\neg\neg r}{\Gamma, \varphi \vdash \Delta, \neg\neg\varphi}\,\text{mon}}{\Gamma, \varphi \vdash \Delta}\,\text{cut}$$

Also, note that $\neg r$ for atoms is a derivable rule:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, p \vdash \Delta}{\neg\Delta \vdash \neg\Gamma, \neg p}\,\text{contrap}}{\neg\Delta, \neg\neg p \vdash \neg\Gamma}\,\neg l}{\neg\neg\Gamma \vdash \neg\neg\Delta, \neg\neg\neg p}\,\text{contrap}}{\Gamma \vdash \Delta, \neg p}\,\neg\neg$$

Next, note that $\Gamma \vdash \Delta, \top$ is a derivable axiom:

$$\dfrac{\dfrac{\neg\Delta, \bot \vdash \neg\Gamma}{\neg\neg\Gamma \vdash \neg\neg\Delta, \top}\,\text{contrap}}{\Gamma \vdash \Delta, \top}\,\neg\neg$$

Finally, we show that the equivalence of $\neg\mathbf{?}(\varphi_1, \varphi_2)$ and $\mathbf{?}(\neg\varphi_1, \neg\varphi_2)$ is provable in the calculus, by showing that the following two rules are derivable:

$$\frac{\Gamma, \neg\varphi_1 \vdash \Delta \qquad \Gamma, \neg\varphi_2 \vdash \Delta}{\Gamma, \neg\mathbf{?}(\varphi_1, \varphi_2) \vdash \Delta} \qquad\qquad \frac{\Gamma \vdash \Delta, \neg\varphi_1 \qquad \Gamma \vdash \Delta, \neg\varphi_2}{\Gamma \vdash \Delta, \neg\mathbf{?}(\varphi_1, \varphi_2)}$$

Here are the relevant derivations:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, \neg\varphi_1 \vdash \Delta}{\neg\Delta \vdash \neg\Gamma, \neg\neg\varphi_1}\,\text{contrap}}{\neg\Delta \vdash \neg\Gamma, \varphi_1}\,\neg\neg \qquad \dfrac{\dfrac{\dfrac{\Gamma, \neg\varphi_2 \vdash \Delta}{\neg\Delta \vdash \neg\Gamma, \neg\neg\varphi_2}\,\text{contrap}}{\neg\Delta \vdash \neg\Gamma, \varphi_2}\,\neg\neg}{\neg\Delta \vdash \neg\Gamma, ?(\varphi_1, \varphi_2)}\,?r}{\neg\neg\Gamma, \neg ?(\varphi_1, \varphi_2) \vdash \neg\neg\Delta}\,\text{contrap}}{\Gamma, \neg ?(\varphi_1, \varphi_2) \vdash \Delta}\,\neg\neg$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash \Delta, \neg\varphi_1}{\neg\Delta, \neg\neg\varphi_1 \vdash \neg\Gamma}\,\text{contrap}}{\neg\Delta, \varphi_1 \vdash \neg\Gamma}\,\neg\neg \qquad \dfrac{\dfrac{\dfrac{\Gamma \vdash \Delta, \neg\varphi_2}{\neg\Delta, \neg\neg\varphi_2 \vdash \neg\Gamma}\,\text{contrap}}{\neg\Delta, \varphi_2 \vdash \neg\Gamma}\,\neg\neg}{\neg\Delta, ?(\varphi_1, \varphi_2) \vdash \neg\Gamma}\,?r}{\neg\neg\Gamma \vdash \neg\neg\Delta, \neg ?(\varphi_1, \varphi_2)}\,\text{contrap}}{\Gamma \vdash \Delta, \neg ?(\varphi_1, \varphi_2)}\,\neg\neg$$

It is not difficult to see that if Γ and Δ do not contain ambiguous formulas, then $\Gamma \vdash_c \Delta$ iff $\Gamma \vdash_a \Delta$. This follows directly from the completeness proof for $\vdash_a$ (with respect to the relation $\models_a$), which is given in the appendix.

4. Predicate Logic with Local Ambiguities

It is a straightforward task to lift the propositional logic of the previous section up to the first-order level. Let $\mathcal{L}_Q$ be the language which extends the normal first-order language with binary ambiguity connective $?(\ldots)$. Let $M = \langle D, I\rangle$ be an ordinary first-order model, with domain D and interpretation I and g a variable assignment which maps the variables of $\mathcal{L}_Q$ to D.

Let $[\![t]\!]_{M,g}$ denote the meaning of t with respect to M and g. That is, if t is a variable then $[\![t]\!]_{M,g} = g(t)$, and if t is a constant then $[\![t]\!]_{M,g} = I(t)$. We define

$$\begin{aligned}
M, g \models R(t_1, \ldots, t_n) &\;\Leftrightarrow\; \langle [\![t_1]\!]_{M,g}, \ldots, [\![t_n]\!]_{M,g}\rangle \in I(R)\\
M, g \dashv R(t_1, \ldots, t_n) &\;\Leftrightarrow\; M, g \not\models R(t_1, \ldots, t_n)\\
M, g \models \forall x\varphi &\;\Leftrightarrow\; M, h \models \varphi \text{ for all } h \text{ s.t. } h(y) = g(y) \text{ whenever } y \neq x\\
M, g \dashv \forall x\varphi &\;\Leftrightarrow\; M, h \dashv \varphi \text{ for some } h \text{ with } h(y) = g(y) \text{ whenever } y \neq x
\end{aligned}$$

The truth and falsity conditions for the connective $\bot$, $\neg$, $\wedge$ and $?(\ldots)$ are the same as for the propositional case.

We say that M supports (rejects) φ iff $M, g \models \varphi$ ($M, g \dashv \varphi$) for all assignments g over M. Furthermore, we say that $\Delta \subseteq \mathcal{L}_Q$ is a valid conclusion from $\Gamma \subseteq \mathcal{L}_Q$, $\Gamma \models_{Qa} \Delta$, iff for all first-order models M:

1. if $M \models \gamma$ for all $\gamma \in \Gamma$ then $M \models \delta$ for certain $\delta \in \Delta$, and

2. if $M \models\!\!\!|\, \delta'$ for all $\delta' \in \Delta$ then $M \models\!\!\!|\, \gamma'$ for certain $\gamma' \in \Gamma$.

Note that this definition yields the classical consequence relation for the restriction of the language to formulas that do not contain the connective $?(\ldots)$.

The ambiguity operator $?(\ldots)$ can be used to treat lexical ambiguities and under-specification of the possessive relation in predicate logical translations of natural language examples. Thus, if we assume that the possessive in (4) is ambiguous between two relations R_1 and R_2, we can give it translation (5).

4 *Bill's house is big.*

5 $\exists x(Hx \wedge ?(R_1xb, R_2xb) \wedge Bx)$.

A sequent axiomatization of $\models_{Qa}$ can be obtained by adding the classical sequent rules for $\forall$ to the axiomatization of $\models_a$ as given in the previous section. In these rules, $\varphi[x/c]$ denotes the result of substituting c for all free occurrences of x in φ. x is free in $?(\varphi_1, \varphi_2)$ iff x is free in φ_1 or in φ_2.

Universal Quantification Rules

$$\frac{\Gamma, \varphi[x/c] \vdash \Delta}{\Gamma, \forall x\varphi \vdash \Delta}\,\forall l \qquad\qquad \frac{\Gamma \vdash \Delta, \varphi[x/c]}{\Gamma \vdash \Delta, \forall x\varphi}\,\forall r$$

Condition on $\forall r$: c does not occur in Γ, Δ.

We write $\Gamma \vdash_{Qa} \Delta$ if $\Gamma \vdash \Delta$ can be deduced from the $\vdash_a$–rules together with the two rules above. In the appendix we will show that this logic is complete for the Qa–consequence relation. Soundness is obvious from the soundness result for the propositional case as described in the previous section.

5. Predicate Logic with Scope Ambiguities

The representation of ambiguity discussed above applies without further ado to scope ambiguity, for an expression A which is ambiguous between two scopings A_1 and A_2 can be represented as $?(A_1, A_2)$. Note, however, that such a representation does not represent a preliminary stage in the processing of ambiguity, but rather a *post hoc* reconstruction: the disambiguations are glued together again by means of $?(\ldots)$, a procedure which in a sense re-creates the ambiguity from the disambiguations. That is not quite what we are after.

We would like to represent scope under-specification by means of a notation that allows unscoped operators as ingredients of formulas. This should also allow for a representation that is more concise than merely spelling out the list of all complete disambiguations.

Consider again the ambiguous example sentence (1), repeated here for convenience as (6).

6 *Every boy did not appear.*

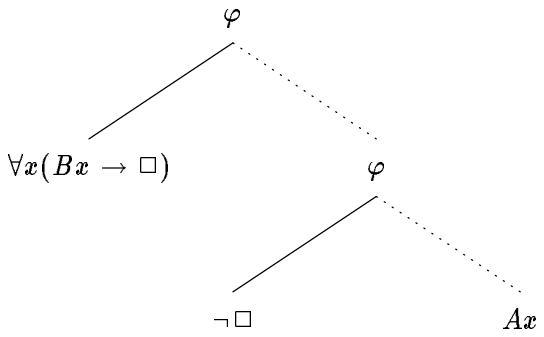

Figure 3: Under-specified predicate logical formula for 'Every boy did not appear.'

In Flexible Montague Grammar, this example gets the following two parses.

7 $[_S\ [_{NP}\ \textit{every boy}]\ [_{VP}\ [_{VP/VP}\ \textit{did not}][_{VP}\ \textit{appear}]]]$

8 $[_S\ [_{NP}\ [_{NP}\ \textit{every boy}]\ [_{NP\backslash NP}\ \textit{did not}]]\ [_{VP}\ \textit{appear}]]$

The constituent *did not* in (7) gets translated as $\lambda P\lambda x.\neg Px$. The type of this translation, $((e,t),(e,t))$, is in accordance with the fact that *did not* acts as a VP modifier. Sentence (7) as a whole gets translation $\forall x(Bx \rightarrow \neg Ax)$. In sentence (8), the constituent *did not* figures as an NP modifier, with translation $\lambda\mathbf{P}\lambda Q.\neg\mathbf{P}(Q)$, and type $(((e,t),t),((e,t),t))$. For the whole sentence, this yields translation $\neg\forall x(Bx \rightarrow Ax)$.

What we have here is two unordered operations and a problem. The unordered operations are universal quantification and negation, and the problem is that these two operations seem to work at different levels: quantification works on predicates and negation works on propositions. If quantification is taken to be an operation from predicates to propositions (as it is in extensional Montague Grammar), then negation has to work on different levels, depending on whether it takes wide or narrow scope. In flexible Montague Grammar this difference shows up as a type shift. For an ambiguous representation with a semantics in terms of unordered operations, the problem of type levels has to be solved.

In predicate logic, the syntactic rule for negation combines a negation operator $\neg$ with a formula. The semantic operation that corresponds to this is taking the complement of the set of variable assignments satisfying the formula. Syntactically, a quantifier $\forall x$ also combines with a formula to yield a new formula. Again, the corresponding operation is an operation on assignment sets, but now a slightly more complex one: take the assignments of which all x variants are in the input set.

Syntactically, we can represent an operator in predicate logic as a formula with a free slot for another formula somewhere inside. Thus, the negation operator can be represented as $\neg\Box$, where $\Box$ is the free formula slot. Similarly, the universal quantification operator translating

'every boy' can be represented as $\forall x(Bx \rightarrow \Box)$. Syntactically, an operator is just a formula with a hole in it. An ambiguous representation of (6) in terms of ambiguous operators can be viewed as a partial tree where a number of operators dominate a formula, but where the domination relation between those operators is left unspecified. See Figure 3 for a representation of (6) in ambiguous predicate logic. The dotted lines indicate the under-specification of the tree dominance relation. It is convenient to define a linear notation for ambiguous predicate logic. An obvious choice is to collect all operators that 'unspecifically' dominate a given formula between parentheses. Thus we arrive at representation (9) for example (6).

9 $(\forall x(Bx \rightarrow \Box), \neg\Box)Ax.$

Turning to a more systematic presentation, the basic ingredients of ambiguous predicate logic are terms (defined as usual), formulas (defined as usual, but with the extra possibility of under-specified operations), operators (essentially formulas containing a free slot $\Box$ for another formula), and contexts (lists of operators). Operators and contexts are to be interpreted as operations on formula-denotations.

$$
\begin{array}{rcl}
t & ::= & c \mid v \\
\varphi & ::= & Rt_1 \cdots t_n \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \vee \varphi_2) \mid \exists v\varphi \mid \forall v\varphi \mid \\
 & & (C_1, \ldots, C_n)\varphi \quad [n \geq 1] \\
\alpha & ::= & \Box \mid \neg\alpha \mid (\alpha \wedge \varphi) \mid (\varphi \wedge \alpha) \mid (\alpha \vee \varphi) \mid (\varphi \vee \alpha) \mid \exists v\alpha \mid \forall v\alpha \mid \\
 & & (C_1, \ldots, C_n)\alpha \quad [n \geq 1] \\
C & ::= & \alpha \mid C\alpha
\end{array}
$$

It is clear from the definition that a formula is either an atom, a negation, a conjunction or a quantification or a list consisting of one or more contexts followed by a formula. Call a formula φ *in explicit form (EF)* if it is not of the form $(C_1, \ldots, C_n)\psi$. A formula in EF does not have an operator scope ambiguity at its top level.

Similarly, an operator is either an EF formula with an open formula slot $\Box$ inside (call these explicit operators), or a list consisting of zero or more contexts followed by an operator (call these complex operators).

Finally, a context is either an operator (a simple context) or a non-singleton list of operators (a complex context).

Instead of $C_1, \ldots, C_n$, we will also write $\widetilde{C_n^1}$. In general, we write $\widetilde{C_j^i}$ for $C_i, C_{i+1}, \ldots, C_{j-1}, C_j$ for every pair of positive numbers i, j with $i \leq j$.

An example of a formula is $(\exists x\Box\forall y\Box, \neg\Box)Rxy$. The part $\neg\Box$ is an example of an operator. The part $\exists x\Box\forall y\Box$ is an example of a context. Contexts may also have a more complex form, such as $(\neg\Box, \exists x\Box)\Box\forall y\Box$. The reason for allowing operator lists (contexts) and operators of the form $(C_1, \ldots, C_n)\alpha$ is to provide a means for representing partial ordering of operations.

10 $((\exists x\Box, \neg\Box)\Box, \forall y\Box)Rxy.$

The representation in (10) expresses that the order of existential quantification and negation with respect to each other is free, and that the order of the universal quantifier with respect to this cluster is free, but that the universal quantifier can not have intermediate scope.

11 $(\exists x \Box \neg \Box, \forall y \Box) R x y$.

The representation in (11) expresses that the order of existential quantification and negation with respect to each other is fixed, but that the universal quantifier can have wide scope, intermediate scope and narrow scope with respect to this cluster.

The substitution of a formula φ for the slot $\Box$ in an operator α, notation $\alpha[\varphi]$, is defined as follows:

$$\begin{array}{rcl}
\Box[\varphi] & := & \varphi \\
(\neg\alpha)[\varphi] & := & \neg\alpha[\varphi] \\
(\alpha \wedge \psi)[\varphi] & := & (\alpha[\varphi] \wedge \psi) \\
(\psi \wedge \alpha)[\varphi] & := & (\psi \wedge \alpha[\varphi]) \\
(\exists v\alpha)[\varphi] & := & (\exists v\alpha[\varphi]) \\
(\forall v\alpha)[\varphi] & := & (\forall v\alpha[\varphi]) \\
(\widetilde{C^1_n}\alpha)[\varphi] & := & \widetilde{C^1_n}(\alpha[\varphi])
\end{array}$$

We use this to define a rewrite relation on formulas (in fact, a step-by-step disambiguation relation), as follows.

$$\begin{array}{rcl}
(\alpha)\varphi & \Rightarrow & \alpha[\varphi] \\
(C\alpha)\varphi & \Rightarrow & (C)\alpha[\varphi] \\
\widetilde{C^1_n}\varphi & \Rightarrow & (\widetilde{C^1_{i-1}}, \widetilde{C^{i+1}_n})\alpha[\varphi] \text{ if } n > 1, C_i = \alpha \\
\widetilde{C^1_n}\varphi & \Rightarrow & (\widetilde{C^1_{i-1}}, C, \widetilde{C^{i+1}_n})\alpha[\varphi] \text{ if } n > 1, C_i = C\alpha
\end{array}$$

For any formula φ let EF(φ) be the following set of formulas:

$$\{\psi \mid \varphi \Rightarrow^* \psi, \text{ with } \psi \text{ in EF}\}$$

It is useful to look at an example of rewriting into explicit form. Note that the rewriting steps can be viewed as disambiguation steps.

$$\begin{array}{rcl}
(\exists x \Box \forall y \Box, \neg\Box) R x y & \Rightarrow & (\exists x \Box, \neg\Box) \forall y R x y \\
& \Rightarrow & (\exists x \Box) \neg \forall y R x y \\
& \Rightarrow & \exists x \neg \forall y R x y
\end{array}$$

$$\begin{array}{rcl}
(\exists x \Box \forall y \Box, \neg\Box) R x y & \Rightarrow & (\exists x \Box, \neg\Box) \forall y R x y \\
& \Rightarrow & (\neg\Box) \exists x \forall y R x y \\
& \Rightarrow & \neg \exists x \forall y R x y
\end{array}$$

$$\begin{aligned}(\exists x\,\Box\forall y\,\Box, \neg\,\Box)Rxy &\Rightarrow (\exists x\,\Box\forall y\,\Box)\neg Rxy\\ &\Rightarrow (\exists x\,\Box)\forall y\neg Rxy\\ &\Rightarrow \exists x\forall y\neg Rxy\end{aligned}$$

The formula $(\exists x\,\Box\forall y\,\Box, \neg\,\Box)Rxy$ does not rewrite to any other formulas in EF. This shows:

$$\mathrm{EF}((\exists x\,\Box\forall y\,\Box, \neg\,\Box)Rxy) = \{\exists x\neg\forall yRxy, \neg\exists x\forall yRxy, \exists x\forall y\neg Rxy\}$$

The notions of freedom and bondage for variables are slightly more subtle here than in standard predicate logic. Intuitively, an occurrence of variable v is 'bound in formula φ' if that occurrence of v ends up bound in every disambiguation of φ, and 'unbound in formula φ' if there is some disambiguation of φ in which it remains free. Here are definitions of the sets of bound and unbound variables that occur in a formula:

$$\begin{aligned}\mathrm{BV}(\varphi) &:= \bigcap\{\mathrm{BV}(\psi) \mid \psi \in \mathrm{EF}(\varphi)\}\\ \mathrm{UV}(\varphi) &:= \bigcup\{\mathrm{UV}(\psi) \mid \psi \in \mathrm{EF}(\varphi)\},\end{aligned}$$

with $\mathrm{BV}(\psi)$ and $\mathrm{UV}(\psi)$ for explicit forms ψ being identical to the definition of bound and unbound occurrences of variables as used in ordinary predicate logic.

A formula φ is a *sentence* (or a *closed formula*) if $\mathrm{UV}(\varphi) = \emptyset$.

Call a variable occurrence *free* in a formula if it remains unbound in every disambiguation of that formula. Note that it may happen that a certain variable occurrence is neither bound nor free in a given formula. Take (12) as an example:

12 $(\exists x(Rxy \wedge \Box), \forall y\Box)Sxy.$

The occurrence of y in Rxy in this formula is neither bound nor free, for there is a disambiguation for (12) where this occurrence remains unbound, viz. $\exists x(Rxy \wedge \forall ySxy)$, and there is also a disambiguation of (12) where the occurrence gets bound, viz. $\forall y\exists x(Rxy \wedge Sxy)$.

The semantics for formulas of ambiguous predicate logic is given in terms of their partial disambiguations in EF. In particular, we have:

$$\begin{aligned}M, g \models (C_1, \ldots, C_n)\varphi &:\Longleftrightarrow M, g \models \psi \text{ for all } \psi \in \mathrm{EF}((C_1, \ldots, C_n)\varphi),\\ M, g \models\!\!\!| (C_1, \ldots, C_n)\varphi &:\Longleftrightarrow M, g \models\!\!\!| \psi \text{ for all } \psi \in \mathrm{EF}((C_1, \ldots, C_n)\varphi).\end{aligned}$$

As can be seen from these clauses, operator scope ambiguity is semantically interpreted as meta choice between all possible permutations of the operations.

As a consequence relation for predicate logic with scope ambiguities we will again take the relation $\models_a$ that was defined in Section 2. Intuitively, this is again plausible. If I maintain that *Every boy did not appear* is true, then it follows at least that some boy did not show up. Conversely, if it is the case that not a single boy showed up, then I can safely maintain that *Every boy did not appear* is true, the ambiguity of my utterance notwithstanding. We can formalize this in the language of ambiguous predicate logic as follows:

13 $(\forall x(Bx \to \Box), \neg\Box)Ax \models_a \neg\forall x(Bx \to Ax)$.

14 $\forall x(Bx \to \neg Ax) \models_a (\forall x(Bx \to \Box), \neg\Box)Ax$.

To give a further formal example of how $\models_a$ works for scope under-specification, consider (15).

15 $(\exists x\Box, \neg\Box)Px \wedge \neg(\exists x\Box, \neg\Box)Px$.

This is under-specified in a situation where nothing is P, for in that situation the two disambiguations of the first conjunct have different truth values, and the two disambiguations of the second conjunct as well, so the conjunction has under-specified truth value. Thus we see that (15) need not always be false.

Similarly, if we mean by a logical truth a formula which is always true, then (16) is not a logical truth.

16 $(\exists x\Box, \neg\Box)Qx \vee \neg(\exists x\Box, \neg\Box)Qx$.

This is under-specified in a situation where nothing is Q, for similar reasons as above.

Now consider the inference from (15) to (16). To see that indeed (15) $\models_a$ (16), we reason as follows. First note that (15) has the special property that there are no models where this formula is true. Similarly, (16) has the special property that there are no models where this formula is false. We now check the two conditions for $\models_a$. Because (15) is not true in any model, condition $M \models (15) \Rightarrow M \models (16)$ holds. Because (16) is not false in any model, the condition $M \models\!\!\!| (16) \Rightarrow M \models\!\!\!| (15)$ holds.

A Gentzen axiomatization

It is straightforward to give a Gentzen style sequent system for predicate logic with scope ambiguities by modifying the Gentzen sequent axiomatization of classical predicate logic, along the lines sketched above for the treatment of $?(\ldots)$.

For the rules of quantification, we need the concept of uniform substitution of a term t for all free occurrences of x in φ. Notation $\varphi[x/t]$. Note that the notion of *free occurrence of a variable* is more subtle here than in standard predicate logic. For instance,

$$((\exists x(Rxy \wedge \Box), \forall y\Box)Sxy)[y/c] = (\exists x(Rxy \wedge \Box), \forall y\Box)Sxy,$$

because no occurrence of y in $(\exists x(Rxy \wedge \Box), \forall y\Box)Sxy$ is free.

We will only state the inference rules that handle ambiguation, and again we will establish soundness as we go along. The sequent rules for ambiguation mirror the definition of the disambiguation relation $\Rightarrow$. Corresponding to the step $(\alpha)\varphi \Rightarrow \alpha[\varphi]$ we have:

Simple Ambiguation

$$\frac{\Gamma, \alpha[\varphi] \vdash \Delta}{\Gamma, (\alpha)\varphi \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, \alpha[\varphi]}{\Gamma \vdash \Delta, (\alpha)\varphi}$$

The following rules correspond to the step $(C\alpha)\varphi \Rightarrow (C)\alpha[\varphi]$:

List Ambiguation

$$\frac{\Gamma, (C)\alpha[\varphi] \vdash \Delta}{\Gamma, (C\alpha)\varphi \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, (C)\alpha[\varphi]}{\Gamma \vdash \Delta, (C\alpha)\varphi}$$

The real ambiguation takes place in the following steps:

$$\begin{aligned} \widetilde{C^1_n}\varphi &\Rightarrow (\widetilde{C^1_{i-1}}, \widetilde{C^{i+1}_n})\alpha[\varphi] \text{ if } n > 1, C_i = \alpha \\ \widetilde{C^1_n}\varphi &\Rightarrow (\widetilde{C^1_{i-1}}, C, \widetilde{C^{i+1}_n})\alpha[\varphi] \text{ if } n > 1, C_i = C\alpha \end{aligned}$$

To find the corresponding rules in the calculus, note that there are n possible single disambiguation steps on $\widetilde{C^1_n}\varphi$, because every single disambiguation step operates on a single context C_i. Write $d_i(\widetilde{C^1_n}\varphi)$ for the result of applying the appropriate disambiguation step on context C_i in formula $\widetilde{C^1_n}\varphi$. In other words:

$$d_i(\widetilde{C^1_n}\varphi) := \begin{cases} (\widetilde{C^1_{i-1}}, \widetilde{C^{i+1}_n})\alpha[\varphi] & \text{if } n > 1, C_i = \alpha \\ (\widetilde{C^1_{i-1}}, C, \widetilde{C^{i+1}_n})\alpha[\varphi] & \text{if } n > 1, C_i = C\alpha \end{cases}$$

Using this abbreviation we can formulate the context ambiguation rules as follows:

Context Ambiguation Left

$$\frac{\Gamma, d_1(\widetilde{C^1_n}\varphi) \vdash \Delta \qquad \cdots \qquad \Gamma, d_n(\widetilde{C^1_n}\varphi) \vdash \Delta}{\Gamma, \widetilde{C^1_n}\varphi \vdash \Delta}$$

Context Ambiguation Right

$$\frac{\Gamma \vdash d_1(\widetilde{C^1_n}\varphi), \Delta \qquad \cdots \qquad \Gamma \vdash d_n(\widetilde{C^1_n}\varphi), \Delta}{\Gamma \vdash \widetilde{C^1_n}\varphi, \Delta}$$

To see the close similarity to the rules for $?(\varphi_1, \varphi_2)$, note that the following sequents are derivable:

$$d_1(\widetilde{C^1_n}\varphi), \ldots, d_n(\widetilde{C^1_n})\varphi \vdash \widetilde{C^1_n}\varphi$$

$$\widetilde{C^1_n}\varphi \vdash d_1(\widetilde{C^1_n}\varphi), \ldots, d_n(\widetilde{C^1_n})\varphi$$

Also, note that if the calculus allows us to derive a sequent in which formula $\widetilde{C^1_n}\varphi$ occurs, then the same sequent, but with $(\widetilde{C^1_{i-1}}, C_{i+1}, C_i, \widetilde{C^{i+2}_n})\varphi$ substituted for $\widetilde{C^1_n}\varphi$, is also derivable.

Here are some example derivations in the calculus. First a derivation of (13):

$$\dfrac{\dfrac{\forall x(Bx \to \neg Ax) \vdash \neg\forall x(Bx \to Ax)}{(\forall x(Bx \to \Box))\neg Ax \vdash \neg\forall x(Bx \to Ax)} \qquad \dfrac{\neg\forall x(Bx \to Ax) \vdash \neg\forall x(Bx \to Ax)}{(\neg\Box)\forall x(Bx \to Ax) \vdash \neg\forall x(Bx \to Ax)}}{(\forall x(Bx \to \Box), \neg\Box)Ax \vdash \neg\forall x(Bx \to Ax)}$$

And here is a derivation of (14):

$$\dfrac{\dfrac{\forall x(Bx \to \neg Ax) \vdash \neg\forall x(Bx \to Ax)}{\forall x(Bx \to \neg Ax) \vdash (\neg\Box)\forall x(Bx \to Ax)} \qquad \dfrac{\forall x(Bx \to \neg Ax) \vdash \forall x(Bx \to \neg Ax)}{\forall x(Bx \to \neg Ax) \vdash (\forall x(Bx \to \Box))\neg Ax}}{\forall x(Bx \to \neg Ax) \vdash (\forall x(Bx \to \Box), \neg\Box)Ax}$$

6. Ambiguating Dynamic Representation Languages

The strategy of adding ambiguous operators also works for other representation languages, such as DPL and DRT. Adding ambiguous operators to DRT, for instance, can be done as follows. Recall the definition of the DRT language:

terms $t ::= v \mid c$

conditions $\varphi ::= \top \mid Pt_1 \cdots t_k \mid v \doteq t \mid v \neq t \mid \neg D \mid D_1 \Rightarrow D_2$

DRSs $D ::= (\{v_1, \ldots, v_n\}, \{\varphi_1, \ldots, \varphi_m\})$

To create an ambiguous version of this, the language can be extended as follows:

terms $t ::= v \mid c$

conditions $\varphi ::= Pt_1 \cdots t_k \mid v \doteq t \mid \neg D \mid D_1 \Rightarrow D_2 \mid (C_1, \ldots, C_n)\varphi$

DRSs $D ::= (\{v_1, \ldots, v_n\}, \{\varphi_1, \ldots, \varphi_m\})$

operators $\alpha ::= \Box \mid \neg E \mid D \Rightarrow E \mid E \Rightarrow D \mid (C_1, \ldots, C_n)E$

openDRSs $E ::= (\{v_1, \ldots, v_n\}, \{\varphi_1, \ldots, \varphi_m, \alpha\})$

contexts $C ::= \alpha \mid C\alpha$

The representation of Figure 1 has the following counterpart in ambiguous DRT:

17 $((\{x\}, \{Bx\}) \Rightarrow \Box), \neg\Box)(\emptyset, \{Ax\})$.

A proof system for an ambiguated version of Dynamic Predicate Logic [9] or DRT [13, 7] would be targeted at a suitable 'dynamized' version of the ambiguous consequence relation $\models_a$ given above. Comparing this with the under-specified DRT of Reyle [19], an obvious difference is that Reyle's under-specified DRT treats ambiguity as object level disjunction between all possible disambiguations. This is unlike the present approach, where ambiguity gives rise to partiality in the semantics, and the extra expressiveness that this yields is exploited in the consequence relation.

7. Conclusion

Ambiguous predicate logic is different from predicate logic with storage (in the spirit of Cooper [5]) and from under-specified representations for predicate logic using metavariables (Muskens [16], Pinkal [17]) in that it makes a genuine proposal for reasoning with ambiguous representations. In that sense it is comparable with proposals to provide the under-specified representations employed in Alshawi c.s. [2] with a semantics of their own: see Alshawi and Crouch [3]. Instead of just proposing a semantics, we also propose a calculus of reasoning with ambiguous expressions. The semantic representation of ambiguity as 'meta-disjunction' bears a certain resemblance to that of Poesio [18], but the consequence relation is quite different.

A difference between the present approach to representing ambiguities and an approach where operators are put in store to be quantified in at a higher level is that just putting operator meanings in store gives less control over intended constraints on scoping than the present representation in terms of ambiguous operators. Consider the well known example (18).

18 *Every man in some city didn't show up.*

Putting the NP meanings for *some city* and *every man in* ... in storage creates the familiar problem of dangling variables, for if the noun phrases are quantified in in the wrong order one ends up with impossible readings such as (19).

19 $\forall x((Mx \wedge Ixy) \rightarrow \exists y(Cy \wedge \neg Sx))$.

The ambiguous operator approach makes it possible to give the sentence an ambiguous representation that excludes this wrong scoping:

20 $(\exists y(Cy \wedge \Box)\forall x((Mx \wedge Ixy) \rightarrow \Box), \neg\Box)Sx$.

Note that the context $\exists y(Cy \wedge \Box)\forall x((Mx \wedge Ixy) \rightarrow \Box)$ fixes the relative scopes of the two noun phrases, but leaves room for intermediate scoping of the negation operator. This gives three possible readings:

21 $\exists y(Cy \wedge \forall x((Mx \wedge Ixy) \rightarrow \neg Sx))$.

22 $\exists y(Cy \wedge \neg\forall x((Mx \wedge Ixy) \rightarrow Sx))$.

23 $\neg\exists y(Cy \wedge \forall x((Mx \wedge Ixy) \rightarrow Sx))$.

Of course, it is not our purpose here to make a linguistic claim about the possible scopings for (18). The purpose of the example is to demonstrate how ambiguous representation can be used to express syntactic constraints on scope ordering, without specifying a fully scoped form.

Indeed, the ambiguous operator representation is reminiscent of a representation of scope ambiguity presented informally, quite long ago, in Kroch [14]. Kroch's account of the creation of scope ambiguities in syntax starts out by collecting operators together in lists determined by surface order.

24 *All of the problems aren't hard to solve.*

25 *Some of the problems aren't hard to solve.*

This would give, respectively, (26) and (27) (in a notation derived from the proposal of Section 5, but with square bracket lists to indicate that order does matter):

26 $[\forall x(Px \rightarrow \Box), \neg\Box]Hx.$

27 $[\exists x(Px \wedge \Box), \neg\Box]Hx.$

Kroch's proposal is to compute the possible scope readings of natural language examples like (24) and (25) on the basis of a series of operator permutations, starting out from surface-structure generated representations like (26) and (27). Some permutations are blocked by so-called output filter conditions, such as: an order of operators $\ldots\neg Q\ldots$ is blocked if Q is the translation of one of the following specifiers: *some, several, a number of, each.* This constraint will allow an operator swap in the case of (26), but forbids the same swap in the case of (27), thus predicting that (24) is ambiguous between the two possible scope orderings of $\forall x(Px \rightarrow \Box)$ and $\neg\Box$, while (25) only has the reading where the negation operator has narrow scope. It would be worth-while to work out a formal version this theory in terms of ambiguous operator representations.

The present proposal moves in a slightly direction from the use of metavariables, as advocated by Muskens and Pinkal. To see how the systems relate, here is an example of ambiguous predicate notation versus metavariable notation for a predicate logical ambiguity.

28 $(\exists x\Box, \neg\Box)Px.$

29 $X_0 = C_1(X_1), X_0 = C_2(X_2), X_1 = (\lambda p.\exists x\ p\ (C_3X_3)), X_2 = (\lambda p.\neg p\ (C_4X_3)), X_3 = Px.$

Because of its generality, the metavariable approach is very promising as a useful representation method for a wide variety of phenomena having to do with under-specification. However, it does not have anything to say as yet about the problem of reasoning with the under-specified structures that it proposes.

At this point the metavariable approach and the present approach supplement each other. A representation that has a well defined level of (under-specified) formulas with a well defined

semantics is much more amenable to the development of reasoning tools for it than a system where the representation of an expression of type *truth value* has to be worked out from a set of equations. Should one be interested in working out the relation of ambiguous consequence for a version of under-specified first order logic written in metavariable notation, then one can always proceed by translating the sets of metavariable equations into ambiguous formulas, use the calculus to reason with these formulas, and then translate back to metavariable representation.

For purposes of natural language semantics preference orderings on possible resolutions of ambiguities would seem useful. For example, in some given context of discourse certain disambiguations for lexical items may be much more plausible than others. An obvious next task, then, is to study reasoning with preferential ambiguity.

8. Appendix: Completeness

In this appendix section we will show the completeness of the propositional logic a and its first-order version Qa.

8.1 Completeness of ambiguous propositional logic

Let $\mathcal{L}$ be the language of propositional logic, and let $\mathcal{L}_a$ be the language of ambiguous propositional logic from Section 3.

Definition 1 *A set* $\Sigma \subseteq \mathcal{L}_a$ *is* a-saturated *if for each* $\Delta \in \mathcal{L}_a$*:*

$$\Sigma \vdash_a \Delta \Rightarrow \Sigma \cap \Delta \neq \emptyset.$$

The definition of saturated set goes back to the completeness techniques as introduced by Aczel [1] and Thomason [22].[3]

Lemma 2 (Lindenbaum lemma for sequents) *If* $\Gamma \nvdash_a \Delta$ *then there exists an* a*-saturated set* Σ *such that* $\Gamma \subseteq \Sigma$ *and* $\Sigma \cap \Delta = \emptyset$*.*

Proof. Lindenbaum construction: $\{\varphi_i\}_{i\in\omega} = \mathcal{L}_a$, $\Sigma_0 = \Gamma$, and $\Sigma_{n+1} = \Sigma_n$ if $\Sigma_n, \varphi_n \vdash_a \Delta$; otherwise $\Sigma_{n+1} = \Sigma_n \cup \{\varphi_n\}$. Take $\Sigma = \lim_{n\to\infty} \Sigma_n$. □

Corollary 3 *If* $\Gamma \nvdash_a \Delta$ *then there exists an* a*-saturated set* Σ *such that* $(\Delta \cup \neg\Gamma) \cap \Sigma = \emptyset$ *and* $\Gamma \subseteq \Sigma$ *or* $\neg\Delta \subseteq \Sigma$*.*

Proof. If $\Gamma \nvdash_a \Delta$ then also $\Gamma \nvdash_a \Delta, \neg\Gamma$ or $\neg\Delta \nvdash_a \Delta, \neg\Gamma$ (this follows from the negation swap rule). Application of Lemma 2 entails the result. □

[3] These authors used this definition for proving the completeness of first-order intuitionistic logic as a modification of the notion of maximal consistency as used in standard Henkin style completeness proofs for classical logic. The sequent style formulation we use here is a generalization of the notion of maximal consistency for classical logic. For a logic which contains the rule $\neg r$, saturation and maximal consistency coincide. This generalisation is a powerful method for proving completeness of logics which are less expressive than classical logic [11, 21].

Definition 4 *Let Σ be an a-saturated set. We define $V_\Sigma : P \longrightarrow \{0,1\}$:*

$$V_\Sigma(p) = \begin{cases} 1 & \text{if } p \in \Sigma \\ 0 & \text{otherwise.} \end{cases}$$

Observation 5

$V_\Sigma(p) = 1 \quad \text{iff} \quad p \in \Sigma$

$V_\Sigma(p) = 0 \quad \text{iff} \quad \neg p \in \Sigma$

Proof. The first item is true by definition. The second item follows from the presence of the rule for left negation introduction for atoms plus the contraposition rule which allows us to derive right negation introduction for atoms. □

Lemma 6 (Truth Lemma) *For all a-saturated sets Σ and all $\varphi \in \mathcal{L}_a$:*

$\neg\varphi \notin \Sigma \;\&\; \varphi \in \Sigma \iff V_\Sigma \models \varphi$ *and*

$\neg\varphi \in \Sigma \;\&\; \varphi \notin \Sigma \iff V_\Sigma \models\!\!\!| \varphi.$

Proof. By induction on the construction of formulas.

$\boxed{p:}$ Observation 5.

$\boxed{\neg\varphi:}$ $V_\Sigma \models \neg\varphi \Leftrightarrow V_\Sigma \models\!\!\!| \varphi \Leftrightarrow \neg\varphi \in \Sigma \;\&\; \varphi \notin \Sigma \Leftrightarrow \neg\varphi \in \Sigma \;\&\; \neg\neg\varphi \notin \Sigma.$

$V_\Sigma \models\!\!\!| \neg\varphi \Leftrightarrow V_\Sigma \models \varphi \Leftrightarrow \neg\varphi \notin \Sigma \;\&\; \varphi \in \Sigma \Leftrightarrow \neg\varphi \notin \Sigma \;\&\; \neg\neg\varphi \in \Sigma.$

$\boxed{\varphi \wedge \psi:}$ $V_\Sigma \models (\varphi \wedge \psi) \Leftrightarrow V_\Sigma \models \varphi \;\&\; V_\Sigma \models \psi \Leftrightarrow \varphi \in \Sigma \;\&\; \neg\varphi \notin \Sigma \;\&\; \psi \in \Sigma \;\&\; \neg\varphi \notin \Sigma \Leftrightarrow \varphi \wedge \psi \in \Sigma \& \neg(\varphi \wedge \psi) \notin \Sigma.$

$V_\Sigma \models\!\!\!| (\varphi \wedge \psi) \Leftrightarrow V_\Sigma \models\!\!\!| \varphi$ or $V_\Sigma \models\!\!\!| \psi \Leftrightarrow \neg\varphi \in \Sigma \;\&\; \varphi \notin \Sigma$ or $\neg\psi \in \Sigma \;\&\; \psi \notin \Sigma \Leftrightarrow \neg(\varphi \wedge \psi) \in \Sigma \;\&\; (\varphi \wedge \psi) \notin \Sigma.$

$\boxed{?(\varphi,\psi):}$ $V_\Sigma \models ?(\varphi,\psi) \iff V_\Sigma \models \varphi \;\&\; V_\Sigma \models \psi \Leftrightarrow \varphi \in \Sigma, \neg\varphi \notin \Sigma, \psi \in \Sigma, \neg\psi \notin \Sigma \Leftrightarrow ?(\varphi,\psi) \in \Sigma, ?(\neg\varphi,\neg\psi) \Leftrightarrow ?(\varphi,\psi) \in \Sigma, \neg?(\varphi,\psi) \notin \Sigma$ (by the interderivability of $?(\neg\varphi,\neg\psi)$ and $\neg?(\varphi,\psi)$).

$V_\Sigma \models\!\!\!| ?(\varphi,\psi) \iff V_\Sigma \models\!\!\!| \varphi \;\&\; V_\Sigma \models\!\!\!| \psi \Leftrightarrow \varphi \notin \Sigma, \neg\varphi \in \Sigma, \psi \notin \Sigma, \neg\psi \in \Sigma \Leftrightarrow ?(\varphi,\psi) \notin \Sigma, ?(\neg\varphi,\neg\psi) \in \Sigma \Leftrightarrow ?(\varphi,\psi) \notin \Sigma, \neg?(\varphi,\psi) \in \Sigma.$

□

Theorem 7 *For all $\Gamma, \Delta \subseteq \mathcal{L}$: $\Gamma \vdash_a \Delta$ iff $\Gamma \models_a \Delta$.*

Proof. Soundness check for $\Rightarrow$ and the usual Lindenbaum argument for $\Leftarrow$: Suppose $\Gamma \not\vdash_a \Delta$. Corollary 3 to Lemma 2 gives an a-saturated set Σ such that either $\Gamma \subseteq \Sigma$ or $\neg\Delta \subseteq \Sigma$, and $\Sigma \cap (\neg\Gamma \cup \Delta) = \emptyset$. This means, according to Lemma 6, that either $V_\Sigma \models \gamma$ and $V_\Sigma \not\models \delta$ for all $\gamma \in \Gamma$ and $\delta \in \Delta$, or $V_\Sigma \models\!\!\!| \delta$ and $V_\Sigma \not\models\!\!\!| \gamma$ for all $\gamma \in \Gamma$ and $\delta \in \Delta$. In either case, $\Gamma \not\models_a \Delta$. □

8.2 Completeness of ambiguous predicate logic

Completeness of the ambiguous predicate logic Qa can be obtained by the same method. The modified Lindenbaum lemma 2 also holds for this system. In fact it holds for all systems which contain the classical structural rules (*start*, *monotonicity* and *cut*) [12]. To obtain the predicate logical results we only need to use *term-saturated sets* instead of ordinary saturated sets.

Definition 8 *A term-Qa-saturated set Σ is a Qa-saturated set in $\mathcal{L}_Q$ such that for every formula $\neg\forall x\varphi x$ there exists a constant c such that $\neg\varphi[x/c] \in \Sigma$.*

Let $\mathcal{L}_Q^*$ be the first-order language which is similar to $\mathcal{L}_Q$ except for the fact that it has a countably infinite number of extra individual constants (these additional constants are also called *parameters*).

Observation 9 *For every Qa-saturated set Σ there exists a term-Qa-saturated set Σ^* in $\mathcal{L}_Q^*$ such that $\Sigma^* \cap \mathcal{L}_Q = \Sigma$.*

Corollary 10 *Let $\Gamma, \Delta \subseteq \mathcal{L}_Q$ such that $\Gamma \not\vdash_{Qa} \Delta$, then there exists a term-Qa-saturated set $\Sigma^* \subseteq \mathcal{L}_Q^*$ such that $\Gamma \subseteq \Sigma^*$ and $\Delta \cap \Sigma^* = \emptyset$.*

Let Σ^* be a term-Qa-saturated set in $\mathcal{L}_Q^*$. The corresponding first-order model M_{Σ^*} consists of a domain D_{Σ^*} which is the set of constants of $\mathcal{L}_Q^*$, and an interpretation I_{Σ^*} which is given by the content of Σ^*: $I_{\Sigma^*}(R) = \{\langle c_1, \ldots, c_n\rangle \in D_{\Sigma^*}^n \mid R(c_1, \ldots, c_n) \in \Sigma^*\}$ and $I_{\Sigma^*}(c) = c$. for all constants c of $\mathcal{L}_Q^*$. Let g be an assignment over M_{Σ^*}, then we define the map $\widehat{g} : \mathcal{L}_Q \longrightarrow \mathcal{L}_Q^*$ to fix the free variables of a given formula according to their g-values in the parametrized language:

$$\widehat{g}(\varphi) := \varphi[x_1/g(x_1), \ldots, x_n/g(x_n)] \text{ where } x_1, \ldots, x_n \text{ is the list of free variables of } \varphi.$$

Observation 11 *For all $\varphi \in \mathcal{L}_Q$, all term-$Qa$-saturated sets Σ^* in $\mathcal{L}_Q^*$ and all assignments g over M_{Σ^*}:*

$$\begin{aligned} \textit{for all } h =_x g : \widehat{h}(\varphi) \in \Sigma^* &\iff \widehat{g}(\forall x\varphi) \in \Sigma^* \\ \textit{for certain } h =_x g : \neg\widehat{h}(\varphi) \in \Sigma^* &\iff \neg\widehat{g}(\forall x\varphi) \in \Sigma^* \end{aligned}$$

This simple but crucial observation ensures the legitimacy of the induction step for the universal quantifier in the relevant truth lemma.

Lemma 12 (Truth Lemma) *Let Σ be a Qa-saturated set in $\mathcal{L}_Q$, and Σ^* be a term-Qa-saturated extension of Σ in $\mathcal{L}_Q^*$. For every $\varphi \in \mathcal{L}_Q$ and for all assignment g over M_{Σ^*}:*

$$\begin{aligned} M_{\Sigma^*}, g \models \varphi &\iff \widehat{g}(\varphi) \in \Sigma^* \textit{ and } \neg\widehat{g}(\varphi) \notin \Sigma^* \\ M_{\Sigma^*}, g \dashv\models \varphi &\iff \widehat{g}(\varphi) \notin \Sigma^* \textit{ and } \neg\widehat{g}(\varphi) \in \Sigma^*. \end{aligned}$$

Proof. The proofs for literals, and the connectives $\bot, \neg, \wedge$ and $?(\ldots)$ are the same as in lemma 6.

$\boxed{\forall x\varphi}$ $M_{\Sigma^*}, g \models \forall x\varphi \Leftrightarrow$ for all $h =_x g : M_{\Sigma^*}, h \models \varphi \Leftrightarrow$ (ind. hyp.)
$\widehat{h}(\varphi) \in \Sigma^*$ and $\neg\widehat{h}(\varphi) \notin \Sigma^*$ for all $h =_x g \Leftrightarrow$ (Obs. 11)
$\widehat{g}(\forall x\varphi) \in \Sigma^*$ and $\neg\widehat{g}(\forall x\varphi) \notin \Sigma^*$.

$M_{\Sigma^*}, g \not\models \forall x\varphi \Leftrightarrow$ for certain $h =_x g : M_{\Sigma^*}, h \not\models \varphi \Leftrightarrow$ (ind. hyp.)
$\widehat{h}(\varphi) \notin \Sigma^*$ and $\neg\widehat{h}(\varphi) \in \Sigma^*$ for certain $h =_x g \Leftrightarrow$ (Obs. 11)
$\widehat{g}(\forall x\varphi) \notin \Sigma^*$ and $\neg\widehat{g}(\forall x\varphi) \in \Sigma^*$. □

Corollary 13 *For all term-Qa-saturated Σ^* in $\mathcal{L}^*_Q$ and for all $\varphi \in \mathcal{L}_Q$:*

$$M^*_\Sigma \models \varphi \iff \varphi \in \Sigma \text{ and } \neg\varphi \notin \Sigma$$
$$M^*_\Sigma \not\models \varphi \iff \varphi \notin \Sigma \text{ and } \neg\varphi \in \Sigma.$$

Theorem 14 *For all $\Gamma, \Delta \subseteq \mathcal{L}_Q$: $\Gamma \vdash_{Qa} \Delta \Leftrightarrow \Gamma \models_{Qa} \Delta$.*

9. Translation into Classical Logic

It is not hard to embed ambiguous propositional and predicate logic into their classical counterparts. We define translations $^+, ^- : \mathcal{L}_a \longrightarrow \mathcal{L}$:

$$\begin{array}{rclrcl} p^+ & = & p & p^- & := & \neg p \\ (\neg\varphi)^+ & = & \varphi^- & (\neg\varphi)^- & := & \varphi^+ \\ (\varphi \wedge \psi)^+ & := & (\varphi^+ \wedge \psi^+) & (\varphi \wedge \psi)^- & := & \neg(\varphi^+ \wedge \psi^+) \\ ?(\varphi,\psi)^+ & := & (\varphi^+ \wedge \psi^+) & ?(\varphi,\psi)^- & := & (\varphi^- \wedge \psi^-) \end{array}$$

Theorem 15 *For each $V : P \longrightarrow \{0,1\}$ and each $\varphi \in \mathcal{L}_a$:*

$$V \models \varphi \Longleftrightarrow V \models \varphi^+ \Longleftrightarrow V(\varphi^+) = 1 \Longleftrightarrow V(\varphi^-) = 0.$$

Example: Let $\varphi := ?(\neg p \wedge q, \neg(p \wedge q))$. Then: $\varphi^+ = \neg p \wedge q \wedge \neg(p \wedge q)$, and $\varphi^- = (p \vee \neg q) \wedge p \wedge q$.

Proof. By a straightforward induction on the construction of formulas. □

Corollary 16 *For all $\Gamma, \Delta \subseteq \mathcal{L}_a$: $\Gamma \models \Delta \Longleftrightarrow \Gamma^+ \models \Delta^+ \;\&\; \Delta^- \models \Gamma^-$.*

The analogous result of this translation to classical logic can be obtained for the first-order case by extending the translations functions with $(\forall x\varphi)^+ = \forall x\varphi^+$ and $(\forall x\varphi)^- = \neg\forall x\varphi^+$.

Acknowledgements
Thanks to Kees van Deemter, Paul Dekker, Manfred Pinkal and Massimo Poesio for helpful comments, discussion and inspiration.

References

1. P. Aczel. Saturated intuitionistic theories. In H. Schmidt, K. Schütte, and H. Thiele, editors, *Contributions to Mathematical Logic*, pages 1–13. North Holland, Amsterdam, 1968.

2. H. Alshawi, editor. *The Core Language Engine.* MIT Press, Cambridge Mass, Cambridge, Mass., and London, England, 1992.

3. H. Alshawi and R. Crouch. Monotonic semantic interpretation. In *Proceedings 30th Annual Meeting of the Association for Computational Linguistics*, pages 32–38, 1992.

4. S. Blamey. Partial logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, volume III*, pages 1–70. Reidel, Dordrecht, Dordrecht, 1986.

5. R. Cooper. *Quantification and Syntactic Theory.* Reidel, Dordrecht, Dordrecht, 1983.

6. K. van Deemter. Towards a logic of ambiguous expressions. In K. van Deemter and S. Peters, editors, *Semantic Ambiguity and Underspecification.* CSLI, Stanford CA, 1996.

7. J. van Eijck and H. Kamp. Representing discourse in context. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic in Linguistics.* Elsevier, Amsterdam, to appear.

8. J.E. Fenstad, T. Langholm, and E. Vespren. Representations and interpretations. In M. Rosner and R. Johnson, editors, *Computational Linguistics and Formal Semantics*, Studies in Natural Language Processing, pages 31–95. Cambridge University Press, Cambridge, 1992.

9. J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.

10. H. Hendriks. *Studied Flexibility; Categories and Types in Syntax and Semantics.* PhD thesis, ILLC, Amsterdam, 1993.

11. J.O.M. Jaspars. *Calculi for Constructive Communication: A Study of the Dynamics of Partial States.* PhD thesis, ILLC Amsterdam and ITK Tilburg, 1994. ILLC dissertation series 1994-4 and ITK dissertation series 1994-1.

12. J.O.M. Jaspars. Partial up and down logic. *Notre Dame Journal of Formal Logic*, 36:134–157, 1995.

13. H. Kamp. A theory of truth and semantic representation. In J. Groenendijk et al., editors, *Formal Methods in the Study of Language.* Mathematisch Centrum, Amsterdam, 1981.

14. A.S. Kroch. *The Semantics of Scope in English.* PhD thesis, MIT, 1974.

15. R. Muskens. *Meaning and Partiality.* PhD thesis, University of Amsterdam, 1989.

16. R. Muskens. Order-independence and underspecification. Manuscript, University of Tilburg, 1995.

17. M. Pinkal. Radical underspecification. Manuscript, University of Saarbruecken, 1996.

18. M. Poesio. Semantic ambiguity and perceived ambiguity. Unpublished manuscript, June 1994.

19. U. Reyle. Dealing with ambiguities by underspecification: Construction, representation

and deduction. *Journal of Semantics*, 10:123–179, 1993.

20. R. Smullyan. *First-order logic*. Springer, Berlin, 1968.
21. E.G.C. Thijsse. *Partial Logic and Knowledge Representation*. PhD thesis, University of Tilburg, Delft, 1992.
22. R.H. Thomason. On the strong semantical completeness of the intuitionistic predicate logic. *Journal of Symbolic Logic*, 33(1):1–7, 1968.