Order-independence and Underspecification[*]

Reinhard Muskens

1 Two Desiderata: Order-independence and Underspecification

In standard Montague Semantics we find a very close correspondence between syntactic and semantic rules (the 'Rule-to-Rule Hypothesis'). This is attractive from a processing point of view, as we like to think of syntactic and semantic processing as being done *in tandem*, with information flowing in both directions, from parsing to interpretation and vice versa. The parsing procedure erects the necessary scaffolding for interpretation, while semantics (and via semantics context and world knowledge) ideally rules out wrong parses at an early stage.

Montague Semantics, however, also favours a strictly *bottom up* semantic processing architecture. The principle of Compositionality, which says that the meaning of a mother node is to be computed from the meanings of her daughters, seems to enforce such a bottom up procedure. Since we know that parsing algorithms that make use of *top down* predictions are often much more efficient than those that do not, and since we do not therefore expect human syntactic processing to be strictly bottom up, there is a dilemma. On the one hand, we want interpretation to be *order-independent:* it should not be decided a priori whether we assign meanings in a top-down, a bottom-up, or any other fashion. On the other hand, the building block picture of meaning that the principle of Compositionality has to offer is attractive too, if it were only because it explains why language users seem to be able to construct unlimited numbers of meanings from the finite set they find in the lexicon.

[*]From: Hans Kamp and Barbara Partee, editors, *Context-Dependence in the Analysis of Linguistic Meaning*, Elsevier, 2004. This paper was presented at the Prague meeting of the 1995 Prague-Teinach Workshop. For some of my more recent views on the description theory of meaning that is introduced here see Muskens [2001] and Van Leusen and Muskens [2003].

A second desirable constraint on processing meanings has to do with the many readings that semantic theories normally assign to any given syntactic input and the combinatorial explosion resulting from this multitude of analyses. As the average sentence will naturally contain at least some scope bearing elements, the number of readings of even a short text may well run into the thousands. Poesio [1994], inspired by Lincoln's saying no doubt, gives the example in (1). Since the two conjuncts of this sentence count five scope bearing elements each, there will be $5!*5! = 14400$ permutations of these elements that respect the constraint that conjunctions are scope islands. Not all of these permutations lead to semantically different readings, but the number of readings that are predicted is still immense.

(1) A politician can fool most voters on most issues most of the time, but no politician can fool all voters on every single issue all of the time.

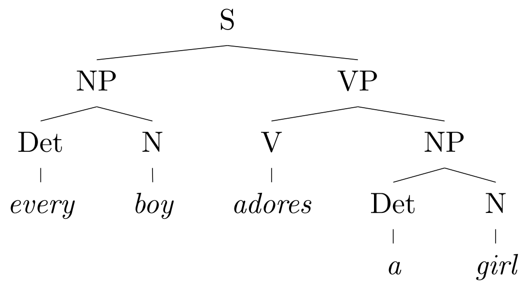
As most texts do not seem to offer insuperable processing difficulties to their readers, it seems improbable that we process sentences by first generating all of their readings and then testing them. Many authors (e.g. Schubert & Pelletier [1982], Alshawi [1992], Reyle [1992], Poesio [1994], Pulman [1994], Cooper et al. [1994], Cooper et al. [1995]) therefore have advocated an approach in which scope ambiguities are not immediately resolved but are allowed to exist for some time. This requires a level of *underspecified representations.* We do not want a representation for each of the multitudinous different scoping possibilities of a given text, but want a single efficient representation for them all. During the parsing process and afterwards, contextual information may narrow down the range of possible scopings of a text; but as long as this process of narrowing down the range of possibilities has not resulted in a unique reading of the text, its semantic representation must be underspecified.

In this paper I shall give a set of simple LFG-like annotated phrase structure rules which connect underspecified representations with other components of the grammar. The rules may be traversed in any order. The underspecified representations will be much like the UDRSs of Reyle [1992], but will formally consist of two parts: (a) a set of equations in ordinary classical type logic with abbreviations that emulate the language of Discourse Representation Theory, and (b) a set of descriptions in the first-order language of trees. Both parts will be generated by the grammar in a relatively independent way. The first part will give us building blocks of 'open' Discourse Representation Structures. The second part will summarise all possible ways in which these building blocks may be combined into an ordinary DRS.

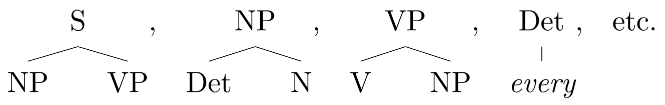
2 Continuous Interpretation in Montague Semantics

In the previous section we have formulated the constraint that the interpretation process should be order-independent. But in fact we may generalise this constraint somewhat and demand that the association of semantic representations with syntactic entities be *continuous.* By this we mean the following. Consider the tree in (2).

(2)



This tree can be thought of as the union of the set of all its local subtrees:



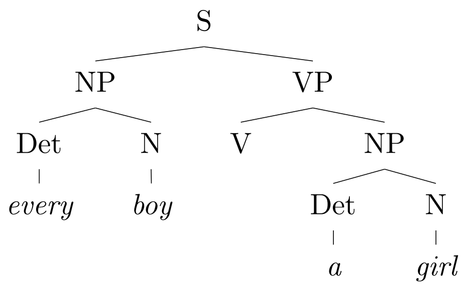
Now supposing that we can find independent interpretations for the local subtrees of (2), the *Continuity Principle* says that the interpretation of (2) itself should simply be the set-theoretic union of those interpretations. The general formulation is as follows:

Principle of Semantic Continuity The interpretation of the union of a set of trees is the union of the interpretations of its elements.

Note that if this principle can be adopted, it will immediately ensure that interpretation can be done order-independently. It will also follow that interpretation is *monotonic,* in the sense that the interpretation of a tree will include the interpretation of any of its subtrees. The principle will moreover allow interpretations to be *partial* in the following sense. Suppose you hear a sentence uttered, but for some reason miss one or more of the words. For instance, suppose that what you actually hear is *Every boy xxxxxx a girl.* Then obviously it will not be possible to interpret the complete utterance. On the other hand, equally obviously, people do assign meanings to what they hear even if they miss some part of it. Supposing that you have managed to assign the incomplete tree (3) to the utterance, the principle of

Continuity will now guarantee that you find some meaning for it, namely the union of the interpretations of all local subtrees of (3).

(3)



In (4) below we have given a description of the tree that is shown in (2). The symbols s_1, np_2, vp_3, etc. are constants referring to nodes. The symbol $\lhd$ denotes the relation of immediate dominance, and we write $a \lhd b,\ c$ as an abbreviation of $a \lhd b \wedge a \lhd c \wedge b \prec c$, where $\prec$ denotes the left-of relation. The function denoted by *cat* assigns a category label to each node and *lex* assigns lexical items to the leaves of a tree. There are of course many trees which satisfy the constraints in (4), but given some reasonable general assumptions about trees (see Backofen et al. 1995 for a first-order axiomatization) the one in (2) is the unique *minimal* tree satisfying the constraints here.

(4)	$s_1 \lhd np_2,\ vp_3$	$cat(s_1) = \mathrm{S}$
	$np_2 \lhd det_4,\ n_5$	$cat(np_2) = \mathrm{NP}$
	$vp_3 \lhd v_6,\ np_7$	$cat(vp_3) = \mathrm{VP}$
	$np_7 \lhd det_8,\ n_9$	$cat(det_4) = \mathrm{Det}$
	$lex(det_4) = every$	$cat(n_5) = \mathrm{N}$
	$lex(n_5) = boy$	$cat(v_6) = \mathrm{V}$
	$lex(v_6) = adores$	$cat(np_7) = \mathrm{NP}$
	$lex(det_8) = a$	$cat(det_8) = \mathrm{Det}$
	$lex(n_9) = girl$	$cat(n_9) = \mathrm{N}$

We can now interpret (2) in the following way. With each node n_i we associate a variable of type logic x_i and we interpret each local subtree with an equation in type logic of the form $x = A$, where x is a variable and A is a term, as in (5) below. The type of a variable follows from its typography: p stands for propositions (type t), P for properties (type et), Q for quantifiers (type $(et)t$), D for type $(et)((et)t)$, V for type $((et)t)(et)$, and x and y for type e.

The interpretation of the entire tree will now simply be the collection of all equations in the second column of (5). It is clear that this system of equations entails that $p_1 = \forall x\,(boy(x) \rightarrow \exists y\,(girl(y) \wedge adores(x)(y))$, but all other indexed variables are likewise equated with a closed term and there are intermediate results as well. Obviously, this way of interpreting conforms to the principle of Continuity and so is order-independent and monotonic. But the principle of Compositionality is respected as well; in fact it is embodied in the first four equations in the second column of (5).

(5)
$$\begin{array}{ll}
s_1 \lhd np_2,\ vp_3 & p_1 = Q_2(P_3) \\
np_2 \lhd det_4,\ n_5 & Q_2 = D_4(P_5) \\
vp_3 \lhd v_6,\ np_7 & P_3 = V_6(Q_7) \\
np_7 \lhd det_8,\ n_9 & Q_7 = D_8(P_9) \\
lex(det_4) = every & D_4 = \lambda P' \lambda P \forall x(P'(x) \rightarrow P(x)) \\
lex(n_5) = boy & P_5 = boy \\
lex(v_6) = adores & V_6 = \lambda Q \lambda y.Q(\lambda x.adores(x)(y)) \\
lex(det_8) = a & D_8 = \lambda P' \lambda P \exists x(P'(x) \wedge P(x)) \\
lex(n_9) = girl & P_9 = girl
\end{array}$$

The interpretation procedure given above is inspired by the way in which f-structures are obtained in Lexical Functional Grammar. Each syntactic rule in LFG comes with functional annotations that give rise to a set of equations constraining f-structure. For example the rule

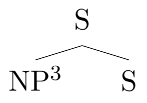
$$\begin{array}{lcc}
\mathrm{S} \rightarrow & \mathrm{NP} & \mathrm{VP} \\
 & \uparrow\mathrm{subj} = \downarrow & \uparrow = \downarrow
\end{array}$$

gives rise to the equations $subj(x_1) = x_2$ and $x_1 = x_3$ saying that the subject attribute of the feature x_1 corresponding with S has the feature x_2 corresponding with the NP as its value and that the features corresponding with S and VP are the same. Lexical items carry similar equations. Thus we can decorate each entry in (4) with a set of feature equations, as it is done in (6) (we suppress category information). The resulting system of equations for the entire c-structure (the *f-description*) describes the f-structure given in (7). The latter, an alternative notation for a certain graph, is the minimal solution of the system of equations in the right column of (6).

(6)	$s_1 \lhd np_2,\ vp_3$	$subj(x_1) = x_2,\ x_1 = x_3$
	$np_2 \lhd det_4,\ n_5$	$x_2 = x_4 = x_5$
	$vp_3 \lhd v_6,\ np_7$	$obj(x_3) = x_7,\ x_3 = x_6$
	$np_7 \lhd det_8,\ n_9$	$x_7 = x_8 = x_9$
	$lex(det_4) = every$	$pred(spec(x_4)) =$ "every", $num(x_4) =$ sg
	$lex(n_5) = boy$	$pred(x_5) =$ "boy", $num(x_5) =$ sg
	$lex(v_6) = adores$	$pred(x_6) =$ "adore", $subj(pred(x_6)) = subj(x_6)$
		$obj(pred(x_6)) = obj(x_6),\ tense(x_6) =$ pres
		$num(subj(x_6)) =$ sg
	$lex(det_8) = a$	$pred(spec(x_8)) =$ "a", $num(x_8) =$ sg
	$lex(n_9) = girl$	$pred(x_9) =$ "girl", $num(x_9) =$ sg

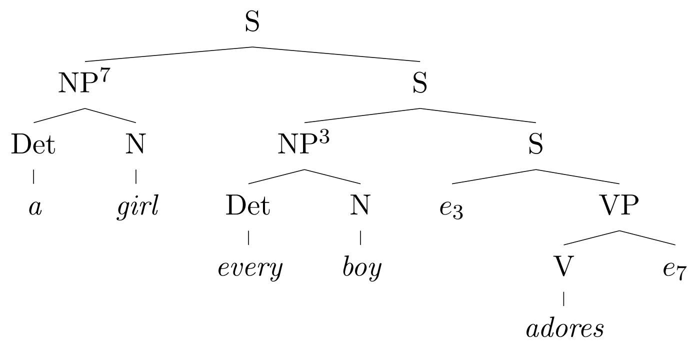
(7)
$$x_1 \left[\begin{array}{lll} \text{PRED} & \multicolumn{2}{l}{\text{``adores}\langle(\uparrow \text{SUBJ}), (\uparrow \text{OBJ})\rangle\text{''}} \\ \text{TENSE} & \text{PRES} & \\ \text{SUBJ} & x_2 & \left[\begin{array}{ll} \text{SPEC} & \left[\begin{array}{ll} \text{PRED} & \text{``every''} \end{array}\right] \\ \text{PRED} & \text{``boy''} \\ \text{NUM} & \text{SG} \end{array}\right] \\ \text{OBJ} & x_7 & \left[\begin{array}{ll} \text{SPEC} & \left[\begin{array}{ll} \text{PRED} & \text{``a''} \end{array}\right] \\ \text{PRED} & \text{``girl''} \\ \text{NUM} & \text{SG} \end{array}\right] \end{array}\right]$$

But the continuous interpretation procedure as illustrated in (5) suffers from a fundamental handicap: it does not seem possible to handle semantic rules that involve *variable binding.* In particular, although the method allows us to treat all *rules of functional application* in Montague's system, it will not allow us to treat his *quantification rules.* Consider the Analysis Tree / Logical Form given in (8). We would like the local subtree



to have its own interpretation, a semantic equation as in (5). But under the usual assumptions there is no way to assign it such an equation with the result that the free variable x_3 corresponding to the trace e_3 gets bound in the solution of the final system of semantic equations. For example, the equation $p_1 = Q(\lambda x_3.p_2)$ (where Q is the variable corresponding to the NP and p_1 and p_2 are variables corresponding to the mother and daughter Ss respectively) will obviously not do. The other equations will enable us to derive that $p_2 = adores(x_3)(x_7)$, but a general condition on substitution ($adores(x_3)(x_7)$ is not free for p_2 in $p_1 = Q(\lambda x_3.p_2)$) prohibits the deduction we would like to make.

(8)



It turns out, however, that the difficulty sketched here need not arise in the dynamic system of Compositional DRT (CDRT), defined in Muskens [1991, 1994, 1996] (for similar systems see Asher & Wada [1989], Groenendijk & Stokhof [1990], Asher [1993], Bos et al. [1994] and Van Eijck & Kamp [1997]), which is a synthesis between Montague Semantics and DRT. The reason is essentially that CDRT is based on a logic (standard type logic) in which a copy of the variable binding mechanism is internalised. This internalisation will allow us to circumvent the present difficulty. In the next section I shall explain how the internalisation is brought about and how CDRT works.

3 Internalising the Binding Mechanism: Compositional DRT

The mathematics underlying CDRT can be explained in two pages. The logic that will be used to internalise binding will be the ordinary three-sorted type logic TY$_3$. Apart from the sort of entities (type e) and the two truth values (type t), we shall also allow for what I would like to call *pigeon-holes* or *registers* (type π) and for *states* (type s). Registers, which are the things that stand proxy for variables and constants, may be thought of as small chunks of space that can contain exactly one object (the value of the variable or constant). States may be thought of as a list of the current inhabitants of all registers. States are very much like the program states that theoretical computer scientists talk about, which are lists of the current values of all variables in a given program at some stage of its execution. Since some registers stand proxy for variables and some for constants, we must distinguish them and we shall have predicates VAR and CON of type π to do so. We shall typically use (subsripted) δ to range over variables and constants of type π; v is a variable of type π; constants u and w (with the exception of w_0) denote registers that model variables; the constant

w_0, but also constants like *John, Mary* or *Sue,* denote registers that stand proxy for constants. Constants u and w denoting variable registers are also called *unspecific* (*discourse*) *referents,* constants denoting constant registers *specific* (*discourse*) *referents.*

In order to be able to impose the necessary structure on our models, we shall let V be some fixed non-logical constant of type $\pi(se)$ and denote the inhabitant of register δ in state i with the type e term $V(\delta)(i)$. We define $i[\delta_1 \ldots \delta_n]j$ to be short for

$$\forall v((\delta_1 \neq v \wedge \ldots \wedge \delta_n \neq v) \rightarrow V(v)(i) = V(v)(j)) \ ,$$

a term which expresses that states i and j differ at most in $\delta_1, \ldots, \delta_n$; $i[\ ]j$ will stand for the formula $\forall v\,(V(v)(i) = V(v)(j))$. We impose the following axioms.

AX1 $\forall i \forall v \forall x(\mathrm{VAR}(v) \rightarrow \exists j(i[v]j \wedge V(v)(j) = x))$

AX2 $\forall i \forall j \forall v(\mathrm{CON}(v) \rightarrow V(v)(i) = V(v)(j))$

AX3 $\mathrm{VAR}(\delta)$, where δ is an unspecific referent

AX4 $\mathrm{CON}(\delta)$, where δ is a specific referent

AX5 $u_n \neq u_m$, for each two different unspecific referents u_n and u_m

AX1 requires that for each state, each variable register and each object, there must be a second state that is just like the first one, except that the given object is an occupant of the given register (this is the 'Having Enough States' axiom from Dynamic Logic, but also the 'Update Axiom' from Janssen's Dynamic Intensional Logic). AX2 says that constant registers have constant values, AX3 and AX4 allow us to make a typographical distinction between constants denoting constant registers and constants denoting variable registers, and AX5 makes sure that different unspecific referents refer to different registers, so that an update on one discourse referent will not result in a change in some other discourse referent's value.

This takes care of the internalisation of the binding mechanism. We now come to the emulation of the DRT language in type logic. Let us agree to write

$$\begin{array}{rll} R\delta_1 \ldots \delta_n & \text{for} & \lambda i.R(V(\delta_1)(i) \ldots V(\delta_n)(i)) \ , \\ \delta_1 \ \mathbf{is} \ \delta_2 & \text{for} & \lambda i.V(\delta_1)(i) = V(\delta_2)(i) \ , \end{array}$$

if R is a term of type $e^n t$ and the δs are constants or variables of type π. This gives us our basic conditions of the DRT language as terms of type st. In order to have complex conditions and boxes as well, we shall write

$$\begin{array}{rll}
\textbf{not}\, K & \text{for} & \lambda i \neg \exists j\, K(i)(j) \ , \\
K \textbf{ or } K' & \text{for} & \lambda i \exists j\, (K(i)(j) \vee K'(i)(j)) \ , \\
K \Rightarrow K' & \text{for} & \lambda i \forall j\, (K(i)(j) \rightarrow \exists k\, K'(j)(k)) \ , \\
[\delta_1 \ldots \delta_n \mid \gamma_1, \ldots, \gamma_m] & \text{for} & \lambda i \lambda j\, (i[\delta_1, \ldots, \delta_n]j \wedge \gamma_1(j) \wedge \ldots \wedge \gamma_m(j)) \ , \\
K \ ; K' & \text{for} & \lambda i \lambda j \exists k\, (K(i)(k) \wedge K'(k)(j)) \ .
\end{array}$$

Here K and K' stand for any term of type $s(st)$, which shall be the type we associate with boxes, and the γs stand for conditions, terms of type st. $[\delta_1 \ldots \delta_n \mid \gamma_1, \ldots, \gamma_m]$ will be our linear notation for standard DRT boxes and the last clause embodies an addition to the standard DRT language: in order to be able to give compositional translations to natural language expressions and texts, we borrow the sequencing operator ';' from the usual imperative programming languages and stipulate that a sequence of boxes is again a box. The following useful lemma is easily seen to hold.

Merging Lemma. If $\vec{u}'$ do not occur in any of $\vec{\gamma}$ then
$\models_{AX} [\vec{u} \mid \vec{\gamma}] \ ; [\vec{u}' \mid \vec{\gamma}'] = [\vec{u}\vec{u}' \mid \vec{\gamma}\vec{\gamma}']$

The present emulation of DRT in type logic should be compared with the semantics for DRT given in Groenendijk & Stokhof [1991]. While Groenendijk & Stokhof give a Tarski definition for DRT in terms of set theory and thus interpret the object DRT language in a metalanguage, the clauses given above are simply abbreviations on the object level of standard type logic. Apart from this difference, the clauses given above and the clauses given by Groenendijk & Stokhof are much the same.

4 Continuous Interpretation of Logical Forms

An important advantage of our emulation of DRT in type logic is that we can now combine the niceties of Montague Semantics with those of Discourse Representation Theory. Most importantly, we can now combine the Montagovian way of composing meanings with the treatment of binding in DRT on the basis of a well-understood and transparant logic. In the translations for a limited set of words given in the table below the constructs of our extended version of DRT are in free combination with lambdas and application. We abbreviate any type of the form $\alpha_1(\ldots(\alpha_n(s(st))\ldots)$ as $[\alpha_1 \ldots \alpha_n]$. Variables p and q are of type [ ], variable P is of type $[\pi]$, and variables v and v' are of type π. Note that all translations are *closed terms.*

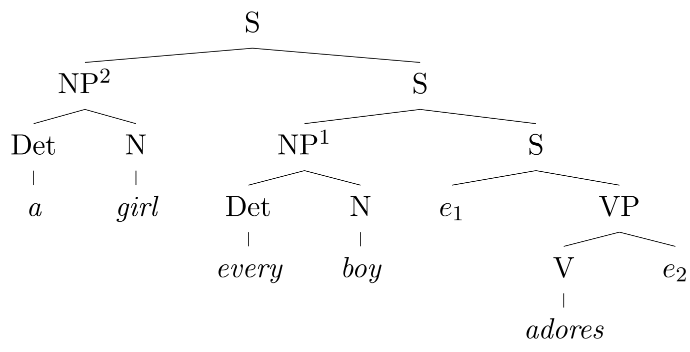
EXPRESSION	TRANSLATION	TYPE
a	$\lambda P \lambda v \lambda p\,([v \mid\,]\;;\;P(v)\;;\;p)$	$[[\pi]\pi[\,]]$
every	$\lambda P \lambda v \lambda p\,[\;\mid ([v \mid\,]\;;\;P(v)) \Rightarrow p]$	$[[\pi]\pi[\,]]$
boy	$\lambda v[\;\mid \textit{boy}\; v]$	$[\pi]$
girl	$\lambda v[\;\mid \textit{girl}\; v]$	$[\pi]$
adores	$\lambda v' \lambda v\,[\;\mid \textit{adores}\; vv']$	$[\pi\pi]$

Let us see how we can use the present system to give a continuous interpretation of a Logical Form which involves some quantification (a monostratal approach will follow in section 6). An example tree is given in (9) below, with its description in the first column of (10). In the third column of (10) each local subtree is paired with a semantic equation, as it was done before. The rules for obtaining semantic equations from local subtrees are as follows. (Variables D are of type $[[\pi]\pi[\,]]$, variables Q of type $[\pi[\,]]$, variables R of type $[\pi\pi]$.)

I Each lexicalised terminal node n_k is paired with an equation $x_k = A$ where A is the translation of the lexical element as in the table above. Each trace e_k with index i is paired with an equation $v_k = u_i$.

II Each local subtree described by $s_k \lhd np_l,\ s_m$ such that np_l is indexed by i is paired with an equation $p_k = Q_l(u_i)(p_m)$. All other local subtrees are interpreted by means of function application.

(9)



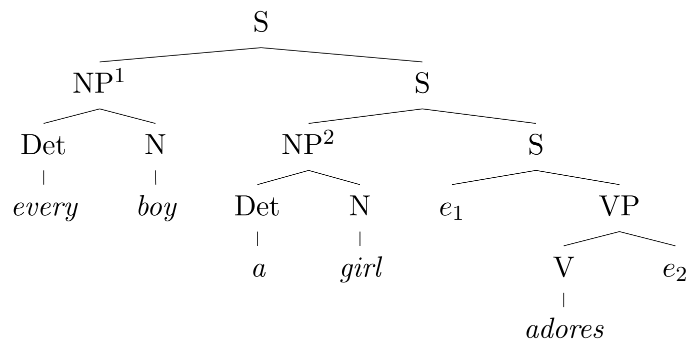
(10)
$$\begin{array}{lll}
s_1 \lhd np_2,\ s_3 & index(np_2) = 2 & p_1 = Q_2(u_2)(p_3) \\
s_3 \lhd np_6,\ s_7 & index(np_6) = 1 & p_3 = Q_6(u_1)(p_7) \\
np_2 \lhd det_4,\ n_5 & & Q_2 = D_4(P_5) \\
np_6 \lhd det_8,\ n_9 & & Q_6 = D_8(P_9) \\
s_7 \lhd e_{10},\ vp_{11} & & p_7 = P_{11}(v_{10}) \\
vp_{11} \lhd v_{12},\ e_{13} & & P_{11} = R_{12}(v_{13}) \\
index(e_{10}) = 1 & & v_{10} = u_1 \\
index(e_{13}) = 2 & & v_{13} = u_2 \\
lex(det_4) = a & & D_4 = \lambda P \lambda v \lambda p\,([v \mid\,] \mathbin{;} P(v) \mathbin{;} p) \\
lex(n_5) = girl & & P_5 = \lambda v[\,\mid girl\ v] \\
lex(det_8) = every & & D_8 = \lambda P \lambda v \lambda p\,[\,\mid ([v \mid\,] \mathbin{;} P(v)) \Rightarrow p] \\
lex(n_9) = boy & & P_9 = \lambda v[\,\mid boy\ v] \\
lex(v_{12}) = adores & & R_{12} = \lambda v' \lambda v\,[\,\mid adores\ vv']
\end{array}$$

The reader may verify that the only variables which occur free in the system of equations in the third column of (10) are the subscripted ones and these are nowhere abstracted over. So we can substitute equals by equals without any restriction. Doing this we find that $p_7 = [\,\mid adores\ u_1u_2]$ is derivable, for example, but also that $p_1 = [u_2 \mid\,] \mathbin{;} [\,\mid girl\ u_2] \mathbin{;} p_3$. The latter can be reduced somewhat further with the help of the Merging Lemma, which allows us to obtain $p_1 = [u_2 \mid girl\ u_2] \mathbin{;} p_3$. In a similar way we get $p_1 = [\,\mid [u_1 \mid boy\ u_1] \Rightarrow p_7]$. The equation corresponding to the top node of the tree hence is

$$p_1 = [u_2 \mid girl\ u_2,\ [u_1 \mid boy\ u_1] \Rightarrow [\,\mid adores\ u_1u_2]]\ .$$

As a second example we may consider the tree in (11). For its description and the system of equations interpreting it, replace the first two rows in (10) by those in (12) (s_3 will now be the maximal S node).

(11)



(12)
$$\begin{array}{lll}
s_3 \lhd np_6,\ s_1 & index(np_6) = 1 & p_3 = Q_6(u_1)(p_1) \\
s_1 \lhd np_2,\ s_7 & index(np_2) = 2 & p_1 = Q_2(u_2)(p_7)
\end{array}$$

The interpretation of (11) is thus summed up by the equations

$$p_3 = [\ |\ [u_1 \mid \mathit{boy}\ u_1] \Rightarrow p_1]$$
$$p_1 = [u_2 \mid \mathit{girl}\ u_2]\ ;\ p_7$$
$$p_7 = [\ |\ \mathit{adores}\ u_1 u_2]$$

From which it is easily derived that

$$p_3 = [\ |\ [u_1 \mid \mathit{boy}\ u_1] \Rightarrow [u_2 \mid \mathit{girl}\ u_2,\ \mathit{adores}\ u_1 u_2]]$$

5 Ambiguous Logical Forms

Let us look again at our two Logical Forms in (9) and (11). They were described by two sets of formulas. In particular (9) was described by

$$s_1 \lhd np_2,\ s_3 \qquad s_3 \lhd np_6,\ s_7$$

plus some other formulas, while (11) was described by

$$s_3 \lhd np_6,\ s_1 \qquad s_1 \lhd np_2,\ s_7$$

plus the same set of other formulas. The relation between the given sets of formulas and the trees is that between a logical theory (set of sentences) and its minimal models. Now if we want to have a representation for both trees at the same time, we must find a theory that has exactly these two trees as its minimal models. Such a theory will be hard to find if we confine ourselves to the language we have been using thus far (unless of course we resort to using disjunctions), but if we take an idea from Marcus et al. [1983] (see also Vijay-Shanker [1992]) and allow the language to talk not only about the relation of *immediate dominance* but also about its reflexive transitive closure (*dominance*), it can be straightforwardly done. In the following set of sentences the symbol $\lhd^*$ denotes dominance.

$$s_{max} \lhd^* s_1 \quad s_1 \lhd np_2,\ s' \quad s' \lhd *s_7$$
$$s_{max} \lhd^* s_3 \quad s_3 \lhd np_6,\ s'' \quad s'' \lhd *s_7$$

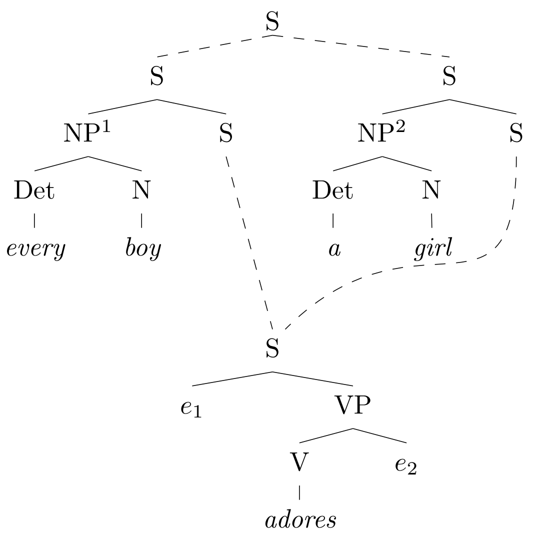
Assuming that np_2 and np_6 cannot denote the same object (as they carry different indices this will be ruled out automatically), there are two minimal models of this theory that conform to the general axioms for binary branching trees, one in which $s_{max} = s_1$, $s' = s_3$ and $s'' = s_7$, and another in which $s_{max} = s_3$, $s'' = s_1$ and $s' = s_7$. Essentially, these two minimal models are

(9) and (11). In (13) we find a complete underspecified representation for these two logical forms. The third column again gives a system of equations for the *Ambiguous Logical Form,* as we shall baptise the description in the first and second columns. Clearly, the interpretation can only be a partial one this time, as there are no obvious semantical analogues to the relation of dominance.

(13)	$s_{max} \lhd^* s_1$		
	$s_{max} \lhd^* s_3$		
	$s' \lhd *s_7$		
	$s'' \lhd *s_7$		
	$s_1 \lhd np_2,\ s'$	$index(np_2) = 2$	$p_1 = Q_2(u_2)(p')$
	$s_3 \lhd np_6,\ s''$	$index(np_6) = 1$	$p_3 = Q_6(u_1)(p'')$
	$np_2 \lhd det_4,\ n_5$		$Q_2 = D_4(P_5)$
	$np_6 \lhd det_8,\ n_9$		$Q_6 = D_8(P_9)$
	$s_7 \lhd e_{10},\ vp_{11}$		$p_7 = P_{11}(v_{10})$
	$vp_{11} \lhd v_{12},\ e_{13}$		$P_{11} = R_{12}(v_{13})$
	$index(e_{10}) = 1$		$v_{10} = u_1$
	$index(e_{13}) = 2$		$v_{13} = u_2$
	$lex(det_4) = a$		$D_4 = \lambda P \lambda v \lambda p\,([v \mid\ ]\ ;\ P(v)\ ;\ p)$
	$lex(n_5) = girl$		$P_5 = \lambda v[\ \mid girl\ v]$
	$lex(det_8) = every$		$D_8 = \lambda P \lambda v \lambda p\,[\ \mid ([v \mid\ ]\ ;\ P(v)) \Rightarrow p]$
	$lex(n_9) = boy$		$P_9 = \lambda v[\ \mid boy\ v]$
	$lex(v_{12}) = adores$		$R_{12} = \lambda v' \lambda v\,[\ \mid adores\ vv']$

Unwieldy representations such as the one in (13) are to be avoided of course and the characteristics of this Ambiguous Logical Form can be conveniently summed up in the *quasi-tree* (the term is Vijay-Shanker's) in (14). Here dashed lines stand for dominance. It should be kept in mind that Ambiguous Logical Forms such as the one in (14) are not trees themselves but stand for descriptions of sets of trees.

(14)



Let us take stock. The interpretation part of (13) is again best summed up by a reduced system of equations as in the second column of (15). The relevant part of the description of the immediate dominance relation is given in the first column.

(15)	$s_{max} \lhd^* s_1$	$p_3 = [\ \mid [u_1 \mid \mathit{boy}\ u_1] \Rightarrow p'']$
	$s_{max} \lhd^* s_3$	$p_1 = [u_2 \mid \mathit{girl}\ u_2]\ ;\ p'$
	$s' \lhd *s_7$	$p_7 = [\ \mid \mathit{adores}\ u_1 u_2]$
	$s'' \lhd *s_7$	
	$s_1 \lhd s'$	
	$s_3 \lhd s''$	

Clearly, by a route different from the one taken in Reyle [1992], we have arrived at an example of an *Underspecified Discourse Representation Structure* (UDRS). It seems that Reyle's UDRSs naturally emerge from a continuous interpretation of Ambiguous Logical Forms. Note, however, that our UDRSs essentially are expressions in a syntactically sugared classical logic plus some information about admissable substitutions of one expression for another.

There are many ways to monotonically add information to (13) so that it admits of only one tree as a minimal model. The addition of $s_{max} = s_1$, for example, will do. On the basis of (13), this sentence, minimality, and general assumptions about trees, it can be inferred that $s' = s_3$ and $s'' = s_7$. Assuming that the latter identities correspond to the identities $p' = p_3$ and $p'' = p_7$, the reading where *every boy* gets wide scope will be obtained.

6 A Grammar Yielding Underspecified Representations

In the previous section we have presented Ambiguous Logical Forms as certain zero order descriptions (sets of literals, to be precise) such that more than one Logical Form could possibly be a minimal tree model of such a description. The question now is how to generate such descriptions. Obviously, it will not do to generate a set of Logical Forms first and then somehow extract the strongest description such that all trees in the set satisfy that, as our desire not to have to generate all possible readings of a text was our motivation for considering Ambiguous Logical Forms in the first place. Instead, we shall give a toy grammar which is meant to illustrate how our variants of Underspecified Discourse Representation Structures can be generated directly.

The annotated phrase structure grammar, which is given in Table 1, is not unlike the grammars that are employed in LFG. The idea is that the grammar not only generates c-structures and f-structures via sets of c-descriptions and f-descriptions, but that it also simultaneously generates the semantic equations (*s-descriptions*) we have met before and a set of *l-descriptions* which partly determine scoping possibilities. In Table 1 we do not show how f-descriptions are being generated, as this is a familiar process. The l-descriptions will not be unlike the sentences in the first column of (15) and thus say something about the Logical Form of the expression, while c-structure is just surface structure. The elements of an l-structure are called *labels.*

We shall use the convention that each non-terminal in a rewrite rule corresponds to a constant in the set of l-descriptions and to a variable in the set of s-descriptions. $l_{\#0}$ denotes the l-description constant corresponding to the mother category, $l_{\#1}$ the constant corresponding to the leftmost daughter, $l_{\#2}$ that corresponding to the second daughter, etc. Similarly, a variable $x_{\#0}$ in the s-description corresponds to the mother node of the local tree covered by the rule, $x_{\#1}$ to the leftmost daughter, etc. Rules may also introduce constants and variables that do not directly correspond to nodes in c-structure (e.g rule (7) introduces l_k, l_m, p_k and p_m). These must be instantiated by fresh constants and variables. b is a function which sends a label corresponding to a c-structure node to the label corresponding to the nearest S node dominating that node in c-structure and in a similar fashion i sends labels to the nearest dominating label that corresponds to a quantifier scope island.

We assume that relative clauses and coordinations give rise to scope islands, but that sentence complement constructions do not. The equation

PS Rules		l-descriptions (+ l-s correspondences)	s-descriptions
(1)	T → S	$l_{\#1} = i(l_{\#1})$	$[w \mid w\ \mathbf{is}\ w_0]\ ;\ p_{\#1}$
(2)	T → T S	$l_{\#0} \lhd l_{\#1},\ l_{\#0} \lhd l_{\#2},$ $l_{\#1} \neq l_{\#2},\ i(l_{\#2}) = l_{\#2}$	$p_{\#0} = p_{\#1}\ ;\ p_{\#2}$
(3)	S → S′	$l_{\#0} \lhd^* l_{\#1},\ b(l_{\#1}) = l_{\#0},$ $i(l_{\#1}) = i(l_{\#0})$	
(4)	S′ → NP′ VP	$l_{\#1} = l_{\#2} = l_{\#0}$	$p_{\#0} = P_{\#2}(v_{\#1})$
(5)	S′ → S″ S	$l_{\#0} \lhd l_{\#1},\ l_{\#0} \lhd l_{\#2},$ $l_{\#1} \neq l_{\#2},\ i(l_{\#1}) = l_{\#1}$ $i(l_{\#2}) = l_{\#2}$	$p_{\#0} = Z_{\#1}(p_{\#2})$
(6)	S″ → IMP S	$l_{\#2} = l_{\#0}$	$Z_{\#0} = C_{\#1}(p_{\#2})$
(7)	VP → NEG VP′	$b(l_{\#0}) \lhd^* l_k \lhd l_m \lhd^* l_{\#0},$ $i(l_{\#0}) \lhd^* l_k,\ l_{\#2} = l_{\#0}$	$P_{\#0} = P_{\#2}$ $p_k = Z_{\#1}(p_m)$
(8)	VP → VP′	$l_{\#1} = l_{\#0}$	$P_{\#0} = P_{\#1}$
(9)	VP′ → V$_{dt}$ NP′ NP′	$l_{\#2} = l_{\#3} = l_{\#0}$	$P_{\#0} = U_{\#1}(v_{\#2})(v_{\#3})$
(10)	VP′ → V$_t$ NP′	$l_{\#2} = l_{\#0}$	$P_{\#0} = R_{\#1}(v_{\#2})$
(11)	VP′ → V$_{in}$		$P_{\#0} = P_{\#1}$
(12)	VP′ → V$_{pa}$ S	$l_{\#0} \lhd l_{\#2},\ i(l_{\#2}) = i(l_{\#0})$	$P_{\#0} = Y_{\#1}(p_{\#2})$
(13)	NP′ → NP	$i(l_{\#0}) \lhd^* l_{\#1} = l_k \lhd l_m \lhd^* l_{\#0}$	$p_k = Q_{\#1}(u_n)(p_m)$ $v_{\#0} = u_n$
(14)	NP′ → PRO		$v_{\#0} = v_{\#1}$
(15)	NP → DET N	$l_{\#2} = l_{\#0}$	$Q_{\#0} = D_{\#1}(P_{\#2})$
(16)	N → N RC	$l_{\#1} = l_{\#2} = l_{\#0}$	$P_{\#0} = X_{\#1}(P_{\#2})$
(17)	RC → RPRO S	$l_{\#0} \lhd l_{\#2},\ i(l_{\#2}) = l_{\#2}$	$X_{\#0} = V_{\#1}(p_{\#2})$
(18)	X → X CONJ X	$l_{\#0} \lhd l_{\#1},\ l_{\#0} \lhd l_{\#3},$ $l_{\#1} \neq l_{\#3},\ i(l_{\#1}) = l_{\#1}$ $i(l_{\#3}) = l_{\#3}$	$x_{\#0} = C_{\#2}(x_{\#1})(x_{\#3})$

Table 1: Annotated Phrase Structure Rules

$i(l_{\#2}) = l_{\#2}$ in rule (16), for example, captures the fact that the S is a scope island here, while the equation $i(l_{\#2}) = i(l_{\#0})$ in rule (12) merely says that the next scope island dominating the S is the island dominating the VP′. The grammar rules that really let things happen are (3), (7) and (13). Rules (7) and (13) send scope bearing elements afloat, merely demanding that the label corresponding to the nearest scope island (and in the case of (7) also the label of the nearest S) must dominate the label of the scope bearing element and that the latter dominates the label of the node where it was generated. Rule (13) moreover generates a discourse referent that interprets the NP′ (and thus functions as the semantic equivalent of a trace), while this discourse referent is simultaneously used in the interpretation of the quantifying expression. This obviates the need for coindexation of quantifiers and their traces at the level of syntax. Rule (3) creates a possible adjunction site where floating scope elements can land. It makes sure that two labels correspond to what conventionally would have been one S node. Of these labels the first need only dominate the second and the possibility that scope bearing elements intervene is left open. (For this technique see also Vijay-Shanker [1992].)

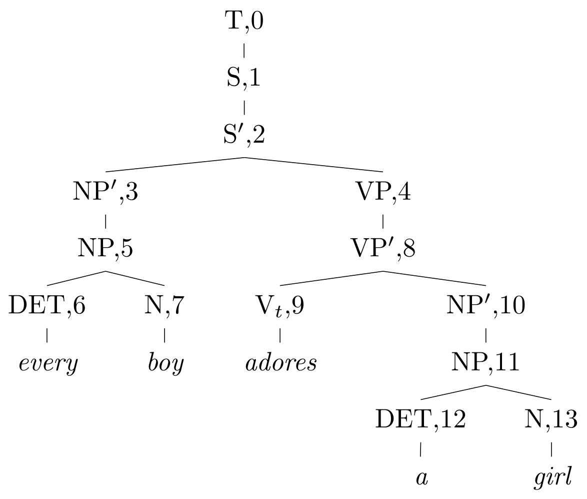
In Table 2 we have given a simple lexicon for the fragment. Since we want to be able to deal with intensional constructions all basic relations are now given an extra argument w, a discourse referent which always stands for the current world of evaluation. The first rule of Table 1 makes sure that this referent is set to the value of w_0, the actual world, at the outset, but later quantifications over worlds in intensional constructions may change this value. In the translation for *believe,* for example, the current world is temporarily stored in referent w' while it is simultaneously required that the value p of the embedded sentence will hold in all worlds w which are doxastic alternatives for the subject v in w' ($Bvww'$ says that w is a doxastic alternative for v in w'). In the translation of the embedded sentence w will again become the current world of evaluation.

We shall assume that some other part of the grammar will coindex relative pronouns and their 'traces' and let these indices play a role in the translation of these elements. With respect to pronouns we follow a different strategy: pronouns will not be indexed themselves, but addition of an equation $v_k = u_i$, where v_k corresponds to a node labelled PRO, will only be allowed if (a) some equation $p = A$ can be shown to hold on the basis of equations that have already been generated and (b) u_i can be shown to be *accessible* from v in A on the basis of some accessibility calculus such as the one in Muskens [1996].

category	expression	translation	type
DET	*a*	$\lambda P \lambda v \lambda p\,([v \mid\,] \;;\; P(v) \;;\; p)$	$[[\pi]\pi[\,]]$
	no	$\lambda P \lambda v \lambda p\,[\,\mid \mathbf{not}([v \mid\,] \;;\; P(v) \;;\; p)]$	$[[\pi]\pi[\,]]$
	every	$\lambda P \lambda v \lambda p\,[\,\mid ([v \mid\,] \;;\; P(v)) \Rightarrow p]$	$[[\pi]\pi[\,]]$
NP	*John*	$\lambda v \lambda p\,([v \mid\; v\ \mathbf{is}\ John] \;;\; P(v))$	$[\pi[\,]]$
N	*girl*	$\lambda v[\,\mid girl\ vw]$	$[\pi]$
V$_{in}$	*stinks*	$\lambda v[\,\mid stinks\ vw]$	$[\pi]$
V$_t$	*adores*	$\lambda v' v[\,\mid adores\ vv'w]$	$[\pi\pi]$
V$_{pa}$	*believes*	$\lambda p \lambda v[w' \mid w'\ \mathbf{is}\ w,\ [w \mid Bvww'] \Rightarrow p]$	$[[\,]\pi]$
RPRO	*who$_i$*	$\lambda p \lambda P \lambda v\,(P(v) \;;\; [u_i \mid v\ \mathbf{is}\ u_i] \;;\; p)$	$[[\,][\pi]\pi]$
NP$'$	*e$_i$*	u_i	π
PRO	*she*	u_i	π
IMP	*if*	$\lambda pq\,[\,\mid p \Rightarrow q]$	$[[\,][\,]]$
NEG	*doesn't*	$\lambda p\,[\,\mid \mathbf{not}\,p]$	$[[\,]]$
CONJ	*and*	$\lambda pq\,(p \;;\; q)$	$[[\,][\,]]$
	or	$\lambda pq\,[\,\mid p\ \mathbf{or}\ q]$	$[[\,][\,]]$

Table 2: The Lexicon

(16)



As a simple example to see how the grammar works, we can have a look at our *every boy adores a girl* sentence again. The grammar will obviously assign the c-structure in (16) to the sentence. We have annotated each non-terminal in this structure with a different number. This can be done in any order. The grammar will connect l-descriptions and s-descriptions to the local trees of this c-structure in the following way. The topmost local tree, which has resulted from the T $\rightarrow$ S rule, will give us descriptions $l_1 = i(l_1)$ and $p_0 = [w \mid w\ \mathbf{is}\ w_0] \;;\; p_1$. The application of S $\rightarrow$ S$'$ gives $l_1 \lhd^* l_2$,

$b(l_2) = l_1$, and $i(l_2) = i(l_1)$. Continuing in this way (a boring task best left to a computer, once understood), we find the descriptions $l_3 = l_4 = l_2$ and $p_2 = P_4(v_3)$ corresponding to the S$'$ $\rightarrow$ NP$'$ VP rule and see that the descriptions $i(l_3) \lhd^* l_5 = l_{14} \lhd l_{15} \lhd^* l_3$, $p_{14} = Q_5(u_1)(p_{15})$ and $v_3 = u_1$ correspond to the leftmost local tree covered by the NP$'$ $\rightarrow$ NP rule. We follow the tree top-down here, but it is clear that any alternative order will give the same results.

In the end, after some very elementary reasoning, we see the following familiar picture emerge: essentially the UDRS that was given in (15).

(17)

$$\begin{array}{ll}
l_1 \lhd^* l_{14} & p_0 = [w \mid w \ \mathbf{is}\ w_0] \ ; p_1 \\
l_1 \lhd^* l_{16} & \\
l_{14} \lhd l_{15} & p_{14} = [\ \mid u_1 \mid \mathit{boy}\ u_1 w] \Rightarrow p_{15}] \\
l_{16} \lhd l_{17} & p_{16} = [u_2 \mid \mathit{girl}\ u_2 w] \ ; p_{17} \\
l_{17} \lhd^* l_2 & p_2 = [\ \mid \mathit{adores}\ u_1 u_2 w] \\
l_{15} \lhd^* l_2 &
\end{array}$$

Disambiguation corresponds to strengthening the l-description, with the understanding that the result must remain consistent with the axioms for unordered trees. As soon as the resulting description entails an equation $l_k = l_m$, we may add the correspondinng equation $p_k = p_m$ to the s-description.

References

[1] Allen, J., 1995, *Natural Language Understanding* (2nd edition), The Benjamins / Cummings Publishing Company, Redwood City, CA.

[2] Alshawi, H. (ed.), 1992, *The Core Language Engine,* MIT Press, Cambridge Mass.

[3] Asher, N., 1993, *Reference to Abstract Objects in Discourse,* Kluwer, Dordrecht.

[4] Asher, N., and H. Wada, 1989, A Computational Account of Syntactic, Semantic and Pragmatic Factors in Anaphora Resolution, *Journal of Semantics.*

[5] Backofen, R., J, Rogers and K. Vijay-Shanker, 1995, A First-Order Axiomatization of the Theory of Finite Trees, *Journal of Logic, Language and Information* **4**, 5-39.

[6] Bos, J., E. Mastenbroek, S. McGlashan, S. Millies and M. Pinkal, 1994, 'A Compositional DRS-based Formalism for NLP Applications', in H. Bunt, R. Muskens and G. Rentier, eds., *Proceedings International Workshop on Computational Semantics,* Institute for Language Technology and Artificial Intelligence, Tilburg, 21-31.

[7] Cooper, R., R. Crouch, J. van Eijck, C. Fox, J. van Genabith, J. Jaspars, H. Kamp, M. Pinkal, M. Poesio, S. Pulman and E Vestre, 1994, *The State of the Art in Computational Semantics: Evaluating the Descriptive Capabilities of Semantic Theories,* FraCaS deliverable D9.

[8] Cooper, R., R. Crouch, J. van Eijck, C. Fox, J. van Genabith, J. Jaspars, H. Kamp, M. Pinkal, M. Poesio, and S. Pulman 1995, *Evaluating the State of the Art,* FraCaS deliverable D10.

[9] Dalrymple, M., J. Lamping and V. Seraswat, 1993, LFG Semantics via Constraints, in *Proceedings of the Sixth European ACL,* Utrecht 1993, 97-105.

[10] Eijck, J. van, and H. Kamp, 1997, 'Discourse Representation Theory', in J.F.A.K van Benthem and A. ter Meulen (eds.), *The Handbook of Logic and Language*, Elsevier Science Publications.

[11] Groenendijk, J. and Stokhof, M., 1990, Dynamic Montague Grammar, in L. Kálmán and L. Pólos (eds.), *Papers from the Second Symposium on Logic and Language,* Akadémiai Kiadó, Budapest, 3-48.

[12] Groenendijk, J. and Stokhof, M.: 1991, Dynamic Predicate Logic, *Linguistics and Philosophy* **14**, 39-100.

[13] Halvorsen, P.-K. and R.M. Kaplan, 1988, Projections and Semantic Description in Lexical-Functional Grammar, in *Proceedings of the International Conference on Fifth Generation Computer Systems,* Tokyo, 1116-1122.

[14] Kaplan, R.M. and J. Bresnan, 1982, Lexical-Functional Grammar: A formal system for grammatical representation, in J. Bresnan (ed.), *The Mental Representation of Grammatical Relations,* The MIT Press, Cambridge, MA, 173-281.

[15] Kaplan, R.M., 1989, The Formal Architecture of Lexical-Functional Grammar, *Journal of Information Science and Engineering* **5**, 305-322.

[16] Leusen, N. van, and Muskens, R., 2003, Construction by Description in Discourse Representation, in J. Peregrin, (ed.), *Meaning: The Dynamic Turn.* Elsevier. 33-67.

[17] Marcus, M.P., D. Hindle and M.M. Fleck, 1983, D-theory: Talking about Talking about Trees, in *Proceedings of the 21st ACL.*

[18] Muskens, R.A., 1991, Anaphora and the Logic of Change, in Jan van Eijck (ed.), *JELIA '90, European Workshop on Logics in AI,* Springer Lecture Notes, Springer, Berlin, 414-430.

[19] Muskens, R., 1994, 'Categorial Grammar and Discourse Representation Theory', *Proceedings of COLING 94,* Kyoto, 508-514.

[20] Muskens, R., 1996, 'Combining Montague Semantics and Discourse Representation', *Linguistics and Philosophy* **19**, 143-186.

[21] Muskens, R., 2001, 'Talking about Trees and Truth-conditions', *Journal of Logic, Language and Information* **10**, 417-455.

[22] Poesio, M., 1994, Ambiguity, Underspecification and Discourse Interpretation, in: H. Bunt, R. A. Muskens and G. Rentier (eds.), *Proceedings of the International Workshop on Computational Semantics,* Tilburg, 151-160.

[23] Pulman, S.G., 1994, A Computational Theory of Context Dependence, in: H. Bunt, R. A. Muskens and G. Rentier (eds.), *Proceedings of the International Workshop on Computational Semantics,* Tilburg, 161-170.

[24] Reyle, U., 1993, Dealing with Ambiguities by Underspecification: Construction, representation and deduction, *Journal of Semantics* **10**, 123-179.

[25] Schubert, L.K. and F.J. Pelletier, 1982, From English to Logic: Context-free computation of conventional logical translation, *American Journal of Computational Linguistics* **8**, 165-176.

[26] Vijay-Shanker, K., 1992, Using Descriptions of Trees in a Tree Adjoining Grammar, *Computational Linguistics* **18**, 481-518.