Talking about Trees and Truth-Conditions*

Reinhard Muskens

To Johan

Abstract

We present *Logical Description Grammar* (LDG), a model of grammar and the syntax-semantics interface based on descriptions in elementary logic. A description may simultaneously describe the syntactic structure and the semantics of a natural language expression, i.e., the describing logic talks about the trees and about the truth-conditions of the language described. Logical Description Grammars offer a natural way of dealing with underspecification in natural language syntax and semantics. If a logical description (up to isomorphism) has exactly one tree plus truth-conditions as a model, it completely specifies that grammatical object. More common is the situation, corresponding to underspecification, in which there is more than one model. A situation in which there are no models corresponds to an ungrammatical input.

1 Introduction

In this paper we present a new model of grammar and the syntax-semantics interface which allows syntactic and semantic representations to be highly underspecified, allows syntactic and semantic information to mutually constrain each other and which potentially allows reasoning on the basis of underspecified forms. The basic idea underlying this model derives from work in computational syntax in the early eighties ((Kaplan and Bresnan 1982), (Marcus, Hindle, and Fleck 1983)) and is sometimes called the *Description Theory* of linguistic representation. It holds, in short, that linguistic representations must in the first place be conceived of as certain *descriptions*

* *Journal of Logic, Language, and Information*,Volume 10, Number 4, Fall 2001.

and can only in a derived sense be understood as whatever is described by those descriptions. One example of this perspective can be found in Lexical-Functional Grammar (Kaplan and Bresnan 1982),[1] where collecting so-called *f-descriptions* preceeds the formation of *f-structures*. A second example is in (Marcus, Hindle, and Fleck 1983), where certain parsing anomalies were avoided by taking descriptions of phrase structure trees, not the trees themselves, as the output of a parsing algorithm. The idea is very general, is essentially theory neutral in the sense that it is compatible with many approaches to grammatical theory, and need not be confined to syntactic theory or parsing. In this paper it will be argued that a similar shift to descriptions is also expedient in semantics. The paper continues work that was done in (Muskens 1995) and (Muskens 1999), but offers an approach that is much simpler, and no less precise, than the treatments found there.

The semantic component of a grammar usually outputs representations which at least determine *truth-conditions*, but we shall move from truth-conditions to descriptions of truth-conditions, just as (Marcus, Hindle, and Fleck 1983) moved from trees to descriptions of trees. In the usual set-up of a semantic theory (the paradigm is (Montague 1973)) a fragment of a natural language is translated into an interpreted logical language. We choose a different approach and let our logical description language essentially function as a metalanguage for English. Our descriptions talk about the truth-conditions of English sentences much in the same way as, in any textbook account of logic, mathematical English talks about the truth conditions of predicate logic. In fact, just as any textbook will define the syntax *and* the semantics of elementary logic, we will *codescribe* trees and truth-conditions—one single logical description will talk about the syntactic and the semantic component of a linguistic object. There seem to be no *a priori* obstacles for also codescribing other levels of the grammar, but these other levels are not considered here. Since we describe linguistic structures with the help of classical logic, we shall call our grammars *Logical Description Grammars* (LDGs).

[1] An even earlier example of the descriptions perspective on syntax are the 'neighbourhood grammars' of (Borščev and Xomjakov 1971). More recent examples are to be found in the 'model-theoretic syntax' of e.g. (Blackburn 1993), (Blackburn, Gardent, and Meyer-Viol 1993), (Rogers 1996), and (Blackburn and Meyer-Viol 1996). The relevance of the descriptions view for the theory of discourse is witnessed in (Gardent and Webber 1998) and (Webber, Knott, and Joshi 1999). A computational perspective, also relevant for the present undertaking, is given in (Duchier and Gardent 1999).

The move from structures and truth conditions to descriptions of structures and their truth conditions offers a uniform and natural way to *underspecify* syntax and semantics. Natural language is replete with syntactic and semantic ambiguities. In (1) we exemplify just a few cases. (1a) is a paradigm illustrating the different possibilities with PP attachment; the real life example (1b) is similar, with a purpose clause that can be attached to different VPs, with different results for the meaning of the whole sentence;[2] and (1c) and (1d) are examples of sentences in which scope ambiguities arise.[3] The reader will have no difficulty to add to these examples.

(1) a. John saw a man with a telescope

b. We do know that there are elements within society here who will take advantage of the quite understandable protest the Orange Order have organised in order to create mayhem.

c. A barbie doll is sold every two seconds somewhere in the world, contributing around two billion dollars a year to Mattel.[4]

d. You may fool all the people some of the time; you can even fool some of the people all the time; but you can't fool all of the people all the time.

The descriptions perspective on language offers an attractive way to model such ambiguity. Since descriptions may be satisfied by more than one model, one representation may correspond to many possibilities and in that case the description compactly represents many readings. This contrasts with a view in which linguistic representations essentially are structures, as this view leads to classifying many structures in terms of one. The last strategy

[2]This sentence is attributed to David Trimble, who is reported to have said it after meeting the Protestant Orange Order at Drumcree in 1998 (BBC News Online, Friday, July, 10, 1998—New push for peace at Drumcree).

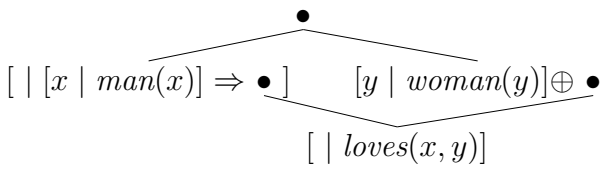
[3](1d) is usually ascribed to Abraham Lincoln, although the historical evidence is somewhat scanty. Note that, while world knowledge makes us interpret (1c) on its $\forall\exists\exists$ reading, the first two clauses of (1d) need not be taken to be fully disambiguated. The first clause can be taken on its $\Diamond\forall\exists$ reading, but the $\Diamond\exists\forall$ option seems also possible. Since the first two clauses of Lincoln's dictum only function rethorically to set the stage for the third, it seems that the exact readings here can be left open to some degree. We conjecture that in such and other instances no disambiguation takes place.

[4]BBC News Online, Tuesday, November 18, 1997—Barbie undergoes plastic surgery.

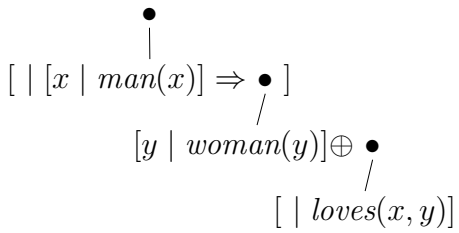
is certainly possible[5] and in semantics is exemplified by the groundbreaking 'Underspecified Discourse Representation Theory' (UDRT) of (Reyle 1993). UDRT essentially generalises existing accounts in Discourse Representation Theory (Kamp 1981; Kamp and Reyle 1993) and its 'Underspecified Discourse Representation Structures' (UDRSs) generalise Discourse Representation Structures. Barring details, a UDRS representation of the ambiguous (2a) looks like the picture in (2b) (here $\oplus$ is the Zeevat merge). Here various segments of Discourse Representation Structures are connected by some ordering $\leq$ which may graphically be represented by a set of lines. The structure in (2b) can be collapsed in two ways. One way of collapsing it leads to (2c) and by a subsequent series of substitutions to the $\forall\exists$ interpretation for the sentence. Another way of collapsing (2b) leads to the $\exists\forall$ reading.

(2) a. Every man loves a woman

b.



c.



Reyle's approach has been very influential. Deservedly so, as it offers a simple picture of how disambiguation works, even if the underlying mathematics in (Reyle 1993) seems a considerable complication of the DRT approach. There are, however, two questions that have to be answered, one technical and one conceptual. The technical question that arises concerns the possibility of the sequence of substitutions that have to be carried out in order to transform a UDRS like (2c) into the ordinary DRS that represents the $\forall\exists$ reading of (2a). In predicate logic, for example, it is certainly not possible to substitute

[5] In a sense feature-based forms of grammar offer classification of structures by structures. A feature structure classifies all those feature structures it subsumes. See (Muskens 1999) for more information on the possibility of feature structures to classify other feature structures.

$\exists y[woman(y) \wedge loves(x,y)]$ for p in $\forall x[man(x) \rightarrow p]$ and obtain the desired capturing of the x in $loves(x,y)$ by the universal quantifier. We can 'solve' this problem by letting forms such as (2b) and (2c) be uninterpreted representations or by employing 'metavariables' (which also shifts the interpretation of the expressions involved), but it would be nicer to interpret ambiguous forms in the same way as their unambiguous counterparts.

The conceptual question—much more important than the technical one—is why a theory that allows for underspecified representations should be a complication of a theory that does not. Ambiguity is not something that should be explained over and above an existing linguistic theory. Ambiguity is an inherent feature of language and should preferably be a direct consequence of the way in which linguistic representation is modeled. A theory which arrives at underspecified representations by means of complicating another theory in which all representations are completely specified seems to predict that language would have been simpler if only it were not ambiguous. In all probability however, language would not have been simpler (but much more complex) if it were not ambiguous.

In the descriptions view that we advocate underspecification naturally follows from the chosen perspective. Indeed, complications of the grammar would be necessary if we were to arbitrarily rule out the possibility that one representation is satisfied by many models. We shall show that the problem with substitution noted above disappears in a pure descriptions approach and we shall also see that representations very much like the one depicted in (2b) appear as a consequence of our theory. However, such representations will not be independently stipulated, but will be epiphenomenal. Our descriptions will be descriptions of ordinary trees and truth-conditions, but can have parts that are not unlike (2b).

The set-up of the rest of the paper is as follows. In the next section we shall informally introduce Logical Description Grammars. Section 3 then gives a formalisation of their syntax and that part of semantics in which the binding of variables plays no role. Section 4 gives some applications and shows how syntactic underspecification naturally follows from the chosen perspective. Section 5, finally, shows how the given semantics can be extended to the case of variable binding. The paper ends with a conclusion.

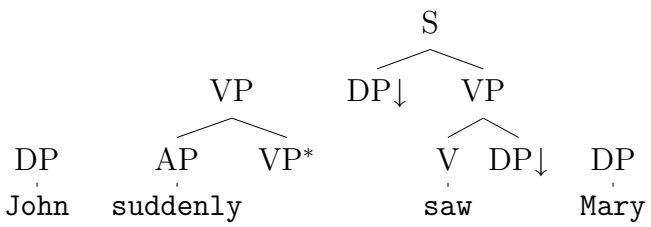
2 The Linguistic Model

The linguistic model underlying Logical Description Grammars derives from (Lexicalised) Tree Adjoining Grammars ((L)TAGs, (Joshi, Levy, and Takahashi 1975; Schabes 1990)) and from D-Tree Grammars (DTGs, (Vijay-Shankar 1992), (Rambow, Vijay-Shanker, and Weir 1995)).[6] The purpose of this section is to give a very informal sketch of the basic idea behind LDGs and their historical continuity (as far as syntax is concerned) with LTAGs and DTGs. The next section will then provide more formal background. We shall give our informal sketch by considering the simple sentence in (3) and analysing it on the basis of the three related theories.

(3) John suddenly saw Mary

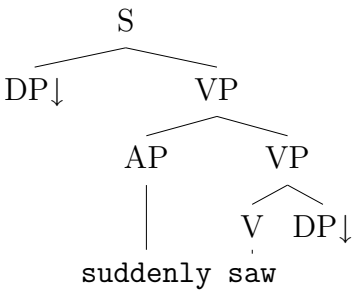
If (3) is analysed with the help of a Lexicalised Tree Adjoining Grammar, first lexical items are associated with each word, so that we arrive at a sequence of *elementary trees* as in (4).

(4)



These elementary trees can then be combined using the operations of *adjoining* and *substitution*. The elementary tree for *suddenly*, for example, can be adjoined to the tree for *saw* at the latter's internal VP node. This then results in the following structure.

(5)

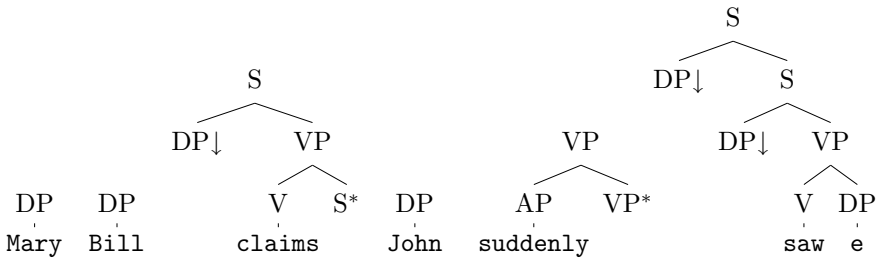


[6]The approach to underspecified semantics taken in (Muskens 1995) was very much inspired by Description Theory and the work in (Vijay-Shankar 1992) but did not offer an actual integration with Tree-Adjoining Grammars. In this paper we endeavour to set this right.

The trees for *John* and *Mary* can subsequently be attached to the two substitution DP nodes (marked with ↓s) and a correct analysis will result.

This very simple example illustrates one of the great attractions of the LTAG framework and also the rudiments of another. The attraction it fully illustrates is the naturalness with which subcategorization is handled. In this example the verb *saw* subcategorizes for a subject and an object DP, which is directly reflected in the geometry of its elementary tree. The other attraction is that LTAGs provide for a mechanism in which unbounded dependencies can be treated at the lexical level. Consider the S node contributed by *saw*. In (4) the distance (measured as the number of immediate dominance edges) between this node and the *saw* node was 3, while in (5) it has become 4 as a result of adjoining the *suddenly* tree. More adjunctions may further increase this distance. This shows that material can 'travel' from its original position and as long as the number of intervening adjunctions is unlimited, the distance that can be travelled is unbounded.

(6)

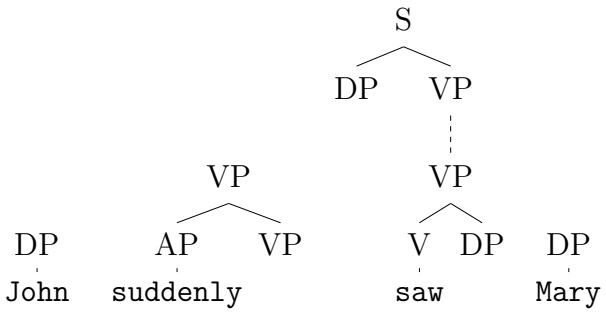


The example in (6) illustrates this further. Here the word *saw* is connected with a variant of the *saw* tree in (4) (words may come with more than one tree). The variant essentially embodies the possibility that extraction has taken place from the object position of the transitive verb. Adjoining the *suddenly* tree to the VP node of the new *saw* tree and adjoining the *claims* tree to the lower S node increases the distance between the 'extracted' DP and its trace. No additional movement operation is necessary. Dependencies (such as case-marking) between extracted element and trace can be regulated on the lexical level and will persist after the movement that is a consequence of tree adjunction has taken place. The lexical tree therefore defines an *extended domain of locality*. In this paper we shall make use of this extended domain of locality for semantic purposes.

Lexicalised Tree Adjoining Grammars definitely offer a structural approach to linguistic representations, but a strongly related descriptions approach was given in the D-Tree Grammar framework of (Vijay-Shankar 1992)

and (Rambow, Vijay-Shanker, and Weir 1995). This work is not only based on LTAGs, but also on the pioneering (Marcus, Hindle, and Fleck 1983), already mentioned in the introduction. As Vijay-Shankar notes in (Vijay-Shankar 1992), the operation of tree adjoining is *destructive* in the sense that it does not preserve tree geometric relations. For example, in (4) the S node in the elementary tree for *saw* is the grandmother of the V node, but in (5) this is no longer so. Within a descriptions approach such destruction can be avoided, as will become clear from example (7). Here the elementary trees are no longer taken to be structures, but are certain descriptions summing up everything that is true in all trees in which the lexical element appears. In the lexical element connected with the word *saw* the single VP node of (4) is replaced by a pair of 'quasi-nodes' which stand in a relation of *domination*, i.e. it is stated that the two VP nodes can be identical or that any positive number of immediate dominance links may intervene. The relation of dominance is indicated with dashed lines while immediate dominance is represented with uninterrupted lines.

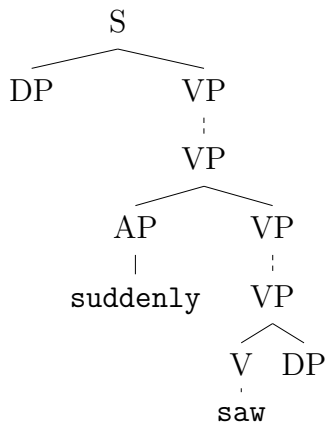
(7)



The correct structure is now obtained by a series of *subsertions*. A subsertion essentially consists of the addition of extra dominance edges. For example, the result of subserting the *suddenly* tree into the *saw* tree is given in (8). This operation essentially adds two dominance relations, one between the VP daughter of S and the mother of AP and one between the sister of AP and the mother of V. Adding two more edges (one between the DP daughter of S and the DP mother of *John* and one between the DP sister of V and the DP mother of *Mary*) results in a D-tree that is already very close to the desired result. The latter is obtained by an operation which removes dominance edges and identifies their end nodes. It is important to note that the whole process can be described in terms of a monotonic increase of information: Subsertion is the addition of dominance edges, the operation

of *sister-adjunction*, which we shall not illustrate here, is a result of similar additions, and the final operation which removes dominance edges in fact strengthens statements about dominance to identity statements.

(8)



While D-Tree Grammars thus make a very important step towards a fully declarative formulation of Tree-Adjoining Grammars they still have an operational flavour. Subsertion and sister-adjunction are two operations on D-trees and the final move of replacing dominance edges by identities is another one. Information must be added in order to arrive at the possible structures and the grammar operationally prescribes how the additions may come about. In contrast, we propose a more radically declarative set-up by characterising the possible phrase structure trees corresponding to a given sentence S simply as the set of models of a certain description $\mathcal{D}$. How $\mathcal{D}$ is composed out of an observable statement about S, a lexicon, and other general information will be explained in the next section. Here we explain the basic intuition behind our set-up. Consider (7) again. There are three kinds of 'quasi-nodes' in the D-trees in this picture.

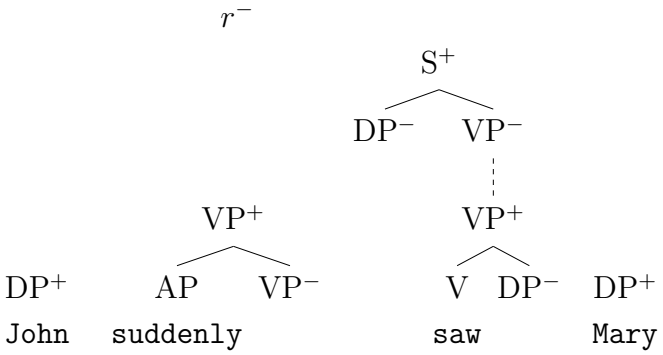
1. Nodes that will never be identified with other nodes, these we call *saturated*.

2. Nodes that may attach to material below them, these we will mark with a $-$.

3. Nodes that may attach to material above them, these we will mark with a $+$.

For example, the VP immediately dominating the V will always be attached to material above it, in any tree in which the lexical item partakes, and is therefore marked +, while the VP daughter of S needs material below it and therefore gets a −. The S node in the elementary tree for *saw* will not get any material above it, as it ends up as the root of the structure here, but in other structures it will have a subordinated position. Therefore it gets a +. In general, the roots of 'components' in DTG are marked positively, while 'substitution nodes' are marked negatively. If nodes are marked in this way we find that the number of + nodes exceeds the number of − nodes by 1. We shall make up for this by adding an extra root node r and marking it negatively. The picture in (9) results.

(9)



(9) is in fact a graphical representation of a description of the original input. Other inputs will lead to similar descriptions. The models of such descriptions can be obtained by pairing off + and − nodes in a one-to-one fashion and identifying the nodes thus paired. I.e. each + node must be identified with a − node and vice versa, but no two +s and no two −s can be identified. The pairing must additionally satisfy the following constraints:

- The result must be a linguistic tree.
- The order of the words must be as given.
- Category information must be respected.

More constraints (e.g. feature information) may of course be added.

There is an obvious close connection between our + and − nodes on the one hand and positive and negative occurrences of types in complex

types in Categorial Grammar (CG).[7] The mechanism that finds models of our descriptions by pairing off node names which are marked + and − can perhaps be called the *chemistry of composition*.[8]

This then is the syntactic part of our linguistic model. Each lexical item consists of a description which we may associate with an elementary tree.[9] The description gives information about phrase structure nodes that are anchored to a certain lexeme. This information not only concerns the tree geometric relations among those nodes and their category labeling, but also whether they are anchored positively or negatively to the lexical item they partake in. Parsing simply consists in pairing off + nodes and − nodes in a way which respects the constraints that were mentioned. In the next section we shall give a simple formalisation of this model in classical logic.

3 Logical Description Grammars Formalised

The graphical representations of descriptions in the previous section are extremely handy and we shall continue using them, but they are used as a shorthand for sentences or sets of sentences in classical logic. The shorthand is not perfect and in this section we explain the 'official' theory and its relation to the graphical forms. We distinguish three kinds of descriptions:

- *general descriptions*, which are nothing but axioms,
- *input descriptions*, which vary per sentence, and
- *lexical descriptions*, which embody the elementary trees of the previous section.

[7] Interestingly, Prof. Guy Perrier of Nancy University in unpublished work independently arrives at a syntactic model strikingly similar to ours by translating proof nets from Lambek Categorial Grammar / Linear Logic in a certain way (Perrier 1998). The similarity between D-Trees and CG is also the subject of (Hepple 1998). In future work we intend to investigate the obvious but fascinating possibility of letting *proof nets* for (Lambek) Categorial Grammars be described by an LDG.

[8] Another useful metaphor comes from the domain of magnetism. Think of positively (negatively) marked constants as of magnets with their positive (negative) poles uncovered. Positive magnets like to combine with negative ones and vice versa, but repulse each other. Saturated nodes are combinations of a positive and a negative magnet and leave no room for other ones.

[9] Or, what is more common, with a set of such trees, among which one must be chosen. We shall consider such ambiguity in the next section.

Each of these will be considered in turn.

3.1 General Descriptions

The structures we are describing are trees and we need axioms to rule out non-trees. The set $\mathcal{A}1$–$\mathcal{A}8$ below (see also (Cornell 1994; Backofen, Rogers, and Vijay-Shankar 1995)) essentially requires the binary relations $\lhd^+$ and $\prec$ to behave like proper dominance and precedence. $\mathcal{A}1$, which uses the abbreviation $k \lhd^* k'$ for $k \lhd^+ k' \vee k = k'$, says that the root r dominates all other nodes; $\mathcal{A}2$–$\mathcal{A}5$ ensure that $\lhd^+$ and $\prec$ are strict partial orderings; $\mathcal{A}6$ is the *Exhaustivity* property that is typical of linguistic trees (nodes are either identical or are in a proper dominance or precedence relation), while $\mathcal{A}7$ and $\mathcal{A}8$ embody the requirement of *Inheritance* (precedence relations are inherited from ancestors).

$\mathcal{A}1$ $\forall k\, r \lhd^* k$

$\mathcal{A}2$ $\forall k \neg k \lhd^+ k$

$\mathcal{A}3$ $\forall k_1 k_2 k_3\, [[k_1 \lhd^+ k_2 \wedge k_2 \lhd^+ k_3] \rightarrow k_1 \lhd^+ k_3]$

$\mathcal{A}4$ $\forall k \neg k \prec k$

$\mathcal{A}5$ $\forall k_1 k_2 k_3\, [[k_1 \prec k_2 \wedge k_2 \prec k_3] \rightarrow k_1 \prec k_3]$

$\mathcal{A}6$ $\forall k_1 k_2\, [k_1 \prec k_2 \vee k_2 \prec k_1 \vee k_1 \lhd^+ k_2 \vee k_2 \lhd^+ k_1 \vee k_1 = k_2]$

$\mathcal{A}7$ $\forall k_1 k_2 k_3\, [[k_1 \lhd^+ k_2 \wedge k_1 \prec k_3] \rightarrow k_2 \prec k_3]$

$\mathcal{A}8$ $\forall k_1 k_2 k_3\, [[k_1 \lhd^+ k_2 \wedge k_3 \prec k_1] \rightarrow k_3 \prec k_2]$

The next two axioms concern the notion of immediate dominance ($\lhd$) and—very incompletely—regulate its relation with $\lhd^+$. $\mathcal{A}9$ states that immediate dominance entails proper dominance, while $\mathcal{A}10$ excludes a possibility that would be in conflict with the notion of immediate dominance being *immediate* dominance.

$\mathcal{A}9$ $\forall k_1 k_2\, [k_1 \lhd k_2 \rightarrow k_1 \lhd^+ k_2]$

$\mathcal{A}10$ $\forall k_1 k_2 k_3 \neg [k_1 \lhd k_3 \wedge k_1 \lhd^+ k_2 \wedge k_2 \lhd^+ k_3]$

In fact, $\mathcal{A}1$–$\mathcal{A}10$ do not suffice to axiomatise the notion of linguistic tree, as they are still satisfied by many undesirable structures. Infinite trees are not ruled out, for example, even though we are only interested in finite ones. Another possibility that should not be allowed to exist is that two nodes could be in a $\lhd^+$ relation without there being a finite path of nodes connected by $\lhd$ between them. The axioms do not exclude this possibility. However, if we choose to rule out such non-intended models with axiomatic means, the exclusion of infinite trees would take us beyond the resources of first-order logic. Below we shall see how a combination of the present axioms with other descriptions will in fact rid us of unintended models.

Nodes in linguistic trees are labeled with category information[10] and we shall assume a domain of *labels*, for which we introduce a special primitive type l. The objects of type l are denoted with label names such as dp, pp, np, etc. The labeling must be functional of course, as (say) a DP can not also be a (say) VP, and we shall express that node n is labeled dp as $\ell(n) = dp$, where ℓ is the labeling function. In order for this to have the intended effect, we also need an axiom scheme to rule out the perverse situation that label names such as dp and vp corefer, i.e. we must require $\mathcal{A}11$.

$\mathcal{A}11$ $c_1 \neq c_2$, if c_1 and c_2 are distinct label names

Instantiations of this axiom scheme will be sentences like $dp \neq vp$ and $ap \neq pp$.

The axioms $\mathcal{A}12$–$\mathcal{A}14$, finally, are part of the machinery driving the chemistry of composition introduced in the previous section. We distinguish between *lexical* nodes (nodes which carry a lexeme) and non-lexical ones and follow the LTAG and DTG approaches by requiring that all nodes should be *anchored*. In fact, we go beyond this and require each node to be both *positively anchored* and *negatively anchored*. The functions α^+ and α^- (of type $\nu\nu$, where ν is the type of tree nodes) give the positive and negative anchors of each node. $\mathcal{A}12$ now requires that the positive anchor of a node is lexical and $\mathcal{A}13$ requires the same for negative anchoring, but exempts the root. The root is negatively anchored to itself ($\mathcal{A}14$).

$\mathcal{A}12$ $\forall k\, lex(\alpha^+(k))$

$\mathcal{A}13$ $\forall k\, [k = r \vee lex(\alpha^-(k))]$

[10]For an easy formulation of the theory we shall identify nodes carrying a terminal string with the first node above them carrying a category symbol.

$\mathcal{A}14\ \ \alpha^{-}(r) = r$

How exactly these axioms, together with other descriptions, help enforce a pairing of positively marked and negatively marked node names will be described shortly.

3.2 Input Descriptions

Input descriptions are constructed on the basis of the observable properties of linguistic objects. In this paper they will all have the same form, which is exemplified by (11), the input description of (3), repeated as (10). The description says that there are four lexical nodes, carrying lexemes *John, suddenly, saw* and *Mary* respectively; that each of these precedes the next; and that the sentence contains no more lexical nodes than these four.

(10) John suddenly saw Mary.

(11) $\exists k_1 k_2 k_3 k_4 (k_1 \prec k_2 \prec k_3 \prec k_4$
$\wedge\ \textsc{john}(k_1) \wedge \textsc{suddenly}(k_2) \wedge \textsc{saw}(k_3) \wedge \textsc{mary}(k_4)$
$\wedge\ \forall k(lex(k) \leftrightarrow (k = k_1 \vee k = k_2 \vee k = k_3 \vee k = k_4)))$

Input descriptions are the only part of our set-up which varies. As will be seen below, the condition which limits the set of lexical nodes plays an important part in ensuring that the models of our descriptions will always be finite.

3.3 Lexical Descriptions

The last set of descriptions we are considering is the *lexicon*. It contains two kinds of descriptions, *classifying descriptions* and *elementary tree descriptions*.

3.3.1 Classifying Descriptions

We need entries classifying *John* and *Mary* as proper names, *suddenly* as an adverb and *saw* as either a transitive verb or a common noun. These entries must also contain information about the semantic values of words. In the classifying descriptions in (12) below, *John* and *Mary* are associated with semantic values of a certain type, while *saw* and *suddenly* are connected with values of other types. The logical translation of *saw* varies, of course, depending on whether the word occurs as a noun or a verb.

(12) a. $\forall k[\textsc{john}(k) \rightarrow (pn(k) \wedge \sigma^{\pi}(k) = john)]$

b. $\forall k[\textsc{mary}(k) \rightarrow (pn(k) \wedge \sigma^{\pi}(k) = mary)]$

c. $\forall k[\textsc{suddenly}(k) \rightarrow (adv(k) \wedge \sigma^{(\pi\tau)(\pi\tau)}(k) = suddenly)]$

d. $\forall k[\textsc{saw}(k) \rightarrow ((cn(k) \wedge \sigma^{\pi\tau}(k) = saw_1) \vee (tv(k) \wedge \sigma^{\pi(\pi\tau)}(k) = saw_2))]$

The typing here is slightly different from what is usual, for reasons that will become apparent below. But the main mechanism behind it is standard. From basic types for names (here: π) and sentences (here: τ) complex types are built in the common way. For example, the type for properties such as saw_1 is $\pi\tau$ and the type for a constant such as *suddenly*, which forms a property if it is given one, is $(\pi\tau)(\pi\tau)$. In (12) we have used our official notation and have employed a family of functions σ^{α} to associate nodes with semantic values of type α, but we shall drop the superscripts immediately for the sake of readability and only use them where we think it enlightening.

3.3.2 Elementary Tree Descriptions

Apart from descriptions that classify words and give semantic translations, such as the ones in (12), the lexicon also has entries which associate substructures of trees with basic word classes. Since these elementary tree descriptions tend to get rather large, we introduce some abbreviations before discussing an example. Firstly, the common combination $k \lhd k' \wedge k \lhd k'' \wedge k' \prec k''$ will be abbreviated as $\Delta(k, k', k'')$. Secondly, the abbreviation $t \overset{+}{\hookleftarrow} \{t_1, \ldots, t_n\}$ will be used to express the information that $t_1, \ldots, t_n$ are exactly the elements that are positively anchored to t. Formally, $t \overset{+}{\hookleftarrow} \{t_1, \ldots, t_n\}$ abbreviates

(13) $\forall k(\alpha^{+}(k) = t \leftrightarrow (k = t_1 \vee \ldots \vee k = t_n))$

The shorthand $t \overset{-}{\hookleftarrow} \{t_1, \ldots, t_n\}$ is used similarly, with α^{-} replacing α^{+}.

Using these abbreviations we can now give the elementary tree description for adverbs.

(14) $\forall k[adv(k) \rightarrow \exists k_2 k_3[\ell(k) = ap \wedge \ell(k_2) = vp \wedge \ell(k_3) = vp \wedge \Delta(k_2, k, k_3) \wedge \sigma(k_2) = \sigma(k)(\sigma(k_3)) \wedge k \overset{+}{\hookleftarrow} \{k, k_2\} \wedge k \overset{-}{\hookleftarrow} \{k, k_3\}]]$

The statement requires nodes that carry an adverb to be labeled AP and have two other VP nodes with them. It also requires certain immediate dominance and precedence relations to hold, and expresses the semantics of one node as that of a second applied to that of a third. Lastly, it strictly circumscribes the nodes that are positively or negatively anchored to the adverb.

Graphical representations such as the ones we have seen earlier are handy for working with elementary tree descriptions. The following conventions will be used for the translation from logic to pictures.

1. Nodes are labeled as usual and node names (variables or constants) are given as subscripts.

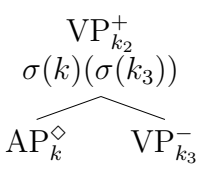
2. Uninterrupted lines stand for immediate dominance ($\lhd$) relations, as usual. Left-right ordering between sisters or between terminal nodes stands for precedence ($\prec$). Dashed lines stand for dominance ($\lhd^*$).

3. Nodes which are positively but not negatively anchored are marked $+$, while nodes which are negatively but not positively anchored are marked $-$. Nodes which are anchored both ways are saturated and unmarked. The situation that a node is not anchored at all will never occur and the anchor itself may be marked with a $\diamond$.

4. The semantic value $\sigma(k)$ of a node k may be written under it.

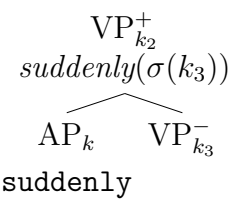
For (14) these rules help us draw the picture (15). The classification of the picture with '*adv:*', means that the description will hold whenever *adv* can be predicated of the anchor k. The solid lines and the ordering between k and k_3 are an alternative to writing $\Delta(k_2, k, k_3)$; writing $\sigma(k)(\sigma(k_3))$ below k_2 is no different from stating $\sigma(k_2) = \sigma(k)(\sigma(k_3))$; etc. (14) can essentially be reconstructed if (15) is given, but some extra readability has been gained. In many situations working with graphical representations such as (15) will be easier than working with the official underlying representation (14).

(15) *adv:*



If (14) is conjoined with the information that k carries the lexeme *suddenly* and with (12c) then the picture in (16) emerges. The semantics of the top VP can be instantiated to *suddenly*$(\sigma(k_3))$.

(16)



The following are elementary tree descriptions for proper names, common nouns and transitive verbs. For the moment we shall stick to the official predicate logical representation.

(17) $\forall k[pn(k) \rightarrow (\ell(k) = dp \wedge k \overset{+}{\hookleftarrow} \{k\} \wedge k \overset{-}{\hookleftarrow} \emptyset)]$

(18) $\forall k[cn(k) \rightarrow (\ell(k) = np \wedge k \overset{+}{\hookleftarrow} \{k\} \wedge k \overset{-}{\hookleftarrow} \emptyset)]$

(19) $\forall k[tv(k) \rightarrow \exists k_2 k_3 k_4 k_5 k_6[\ell(k) = v \wedge \ell(k_2) = s \wedge \ell(k_3) = dp \wedge$
$\ell(k_4) = vp \wedge \ell(k_5) = vp \wedge \ell(k_6) = dp \wedge \Delta(k_2, k_3, k_4) \wedge \Delta(k_5, k, k_6) \wedge$
$k_4 \lhd^* k_5 \wedge \sigma(k_5) = \sigma(k)(\sigma(k_6)) \wedge \sigma(k_2) = \sigma(k_4)(\sigma(k_3)) \wedge$
$k \overset{+}{\hookleftarrow} \{k, k_2, k_5\} \wedge k \overset{-}{\hookleftarrow} \{k, k_3, k_4, k_6\}]]$

3.4 Parsing as Deduction

The hearer of (10) who has concluded (11) and who has the lexical descriptions discussed above in her lexicon may now do some elementary reasoning. She may take witnesses 1, 2, 3, and 4 for the nodes described in (11),[11] which leads to (20), and she may combine this information with (12) in order to obtain (21).

(20) $1 \prec 2 \prec 3 \prec 4 \wedge \textsc{john}(1) \wedge \textsc{suddenly}(2) \wedge \textsc{saw}(3) \wedge \textsc{mary}(4)$
$\wedge \forall k(lex(k) \leftrightarrow (k = 1 \vee k = 2 \vee k = 3 \vee k = 4))$

(21) a. $pn(1) \wedge \sigma(1) = john$

b. $adv(2) \wedge \sigma(2) = suddenly$

c. $(cn(3) \wedge \sigma(3) = saw_1) \vee (tv(3) \wedge \sigma(3) = saw_2)$

[11] Numerals $1, 2, 3, \ldots$ will be used as constants of the node type ν. Clearly, none of the usual properties of $1, 2, 3, \ldots$ (as names for natural numbers) are intended to follow from this usage. Statements such as $4 = 11$ below will be perfectly consistent.

d. $pn(4) \wedge \sigma(4) = mary$

The third item here is a disjunction, which may be eliminated by considering the disjuncts one by one. Let us suppose that our hearer hypothetically assumes (22).

(22) $tv(3) \wedge \sigma(3) = saw_2$

With the help of the information obtained or hypothesized thus far and the elementary tree descriptions in (14)–(19) the following conjunctions can be derived. Here fresh witnesses are taken wherever required.

(23) $\ell(1) = dp \wedge 1 \overset{+}{\hookleftarrow} \{1\} \wedge 1 \overset{-}{\hookleftarrow} \emptyset$

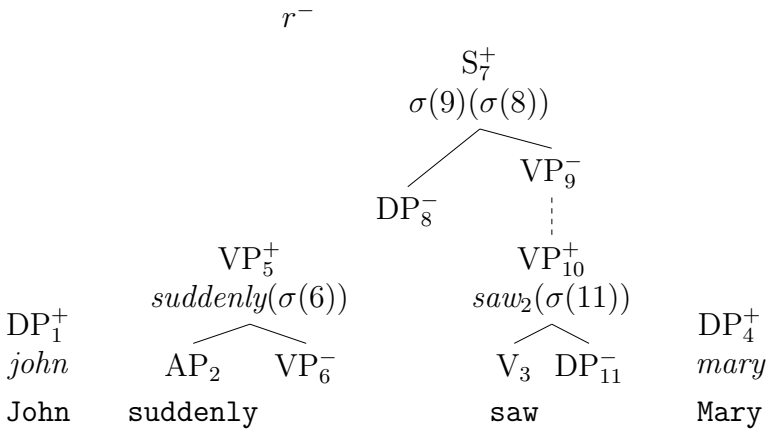
(24) $\ell(2) = ap \wedge \ell(5) = vp \wedge \ell(6) = vp \wedge \Delta(5, 2, 6) \wedge \sigma(5) =$
$suddenly(\sigma(6)) \wedge 2 \overset{+}{\hookleftarrow} \{2, 5\} \wedge 2 \overset{-}{\hookleftarrow} \{2, 6\}$

(25) $\ell(3) = v \wedge \ell(7) = s \wedge \ell(8) = dp \wedge \ell(9) = vp \wedge \ell(10) = vp \wedge$
$\ell(11) = dp \wedge \Delta(7, 8, 9) \wedge \Delta(10, 3, 11) \wedge 9 \lhd^* 10 \wedge$
$\sigma(10) = saw_2(\sigma(11)) \wedge \sigma(7) = \sigma(9)(\sigma(8)) \wedge$
$3 \overset{+}{\hookleftarrow} \{3, 7, 10\} \wedge 3 \overset{-}{\hookleftarrow} \{3, 8, 9, 11\}$

(26) $\ell(4) = dp \wedge 4 \overset{+}{\hookleftarrow} \{4\} \wedge 4 \overset{-}{\hookleftarrow} \emptyset$

We are now in fact at the stage we have previously described informally in (9). In (27) this figure is considered again, with subscripts added to the nodes in order to facilitate comparison with the formal material.

(27)



The series of elementary trees here conveniently sums up the information in (23)–(26) and the input description (11), plus the information (from the axioms) that there must be a root node r which is negatively anchored.

Our hypothetical hearer may now reason as follows. By the axioms for $\prec$ and the precedence information in (20), 1, ..., 4 must be pairwise distinct. All nodes must be positively anchored to a lexical node ($\mathcal{A}12$). Since, by (20), 1, ..., 4 are the only lexical nodes, it follows from (23)–(26) that each node in the domain must be designated by one of 1, 2, 3, 4, 5, 7, or 10. For each distinct pair n, n' of these constants there are two possibilities.

1. Some statement of the form $n'' \overset{+}{\hookleftarrow} A$, with $n, n' \in A$ is one of the conjuncts of one of (23)–(26). Then $n \neq n'$ follows from that lexical description and the tree axioms. Or,

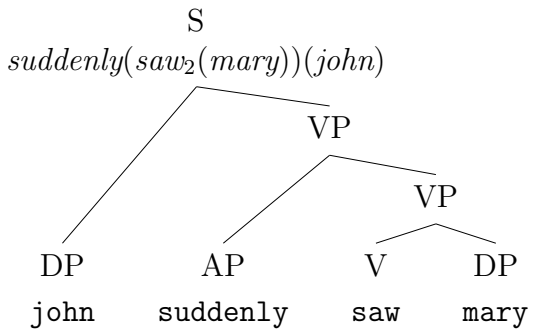
2. it can be derived that n and n' are positively anchored to distinct lexical elements, in which case we also have $n \neq n'$.

So 1, 2, 3, 4, 5, 7, and 10 make up the whole domain of nodes and no two distinct constants in this set can corefer. By a similar argument, but using the information about negative anchoring and what is known about the root, it can be shown that 2, 3, 6, 8, 9, 11, and r also make up the whole domain and are pairwise distinct. It follows that each of 1, 4, 5, 7, and 10 must corefer with exactly one of 6, 8, 9, 11, and r. Given the additional information about labeling and tree structure contained in (20) and (23)–(26) and the constraints imposed upon these in the axioms, there is only one model in which this is the case. In this model it holds that

(28) $1 = 8 \wedge 4 = 11 \wedge 5 = 9 \wedge 6 = 10 \wedge 7 = r.$

This means that we arrive at the following picture in which our tree is completed and the semantics of its root node is computed.

(29)



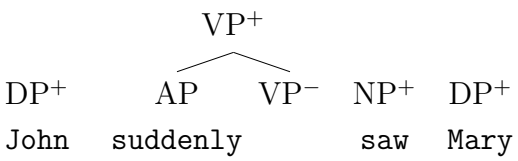
(30) $\sigma(r) = suddenly(saw_2(mary))(john)$

But our hearer is not finished yet. She has derived (30) on the basis of the assumption in (22) and so the alternative possibility (31), in which *saw* is taken to be a common noun, must be considered.

(31) $cn(3) \wedge \sigma(3) = saw_1$

Clearly, this possibility is not satisfiable. It will lead to a description as in (32) below (semantic information is suppressed here). But here there is no one-to-one pairing between positively marked and negatively marked node names, let alone a pairing which respects category, tree and ordering information.

(32)



It can be concluded that the semantical information (30) follows from the original input description (11), given the set of general descriptions and lexical descriptions $\mathcal{G}'$considered thus far. Suppose that we have a grammar $\mathcal{G}$ which is a set of such descriptions. Then, writing $\Gamma \models_{\mathcal{G}} \varphi$ for $\Gamma, \mathcal{G} \models \varphi$, we may conclude that (11)$\models_{\mathcal{G}}$(30) if $\mathcal{G}' \subseteq \mathcal{G}$. It is in this sense that our notion of grammar is purely declarative. We are interested in finding a suitable $\mathcal{G}$ such that, given any string of words of the language, that string is acceptable if and only if the input description connected with the string is consistent with $\mathcal{G}$. Moreover, the linguistically relevant properties of a grammatical string should be $\models_{\mathcal{G}}$ consequences of its input description.

Note that the way in which the semantic information in (30) was obtained is clearly reminiscent of the way in which feature descriptions are collected in Lexical-Functional Grammar. We shall come back to this shortly. Another thing which is to be noted is that all values of σ in the example can be denoted by means of *closed* terms. In section 5 it will be explained that a straightforward attempt to apply the method to sentences with quantifying DPs fails. Quantification essentially involves nodes with open expressions as their translations, or so at least it seems. A straightforward approach then leads to difficulties with substitution, as a logically minded reader may already have noticed. How these hurdles can be overcome will also be explained in section 5.

3.5 Implicit Information

We have seen that the descriptions considered thus far admit only finite models. This was not because some explicit stipulation excluded infinite structures but followed from the way input descriptions and elementary tree descriptions were set up. Input descriptions circumscribe the lexical nodes of the tree as a specific finite set and elementary tree descriptions specify the finite number of elements positively or negatively anchored to any given lexical node. In particular, they always introduce statements $t \overset{+}{\hookleftarrow} \{t_1, \ldots, t_p\}$ and $t \overset{-}{\hookleftarrow} \{t'_1, \ldots, t'_q\}$, where t denotes the lexical element.[12] The trick here is that we do not try to describe once and for all what are the acceptable structures of the language. Since finite structures of arbitrary size are acceptable this is not possible with first-order means.[13] Instead we require our input descriptions to have a certain form. The rest of our descriptions accept a given input just if they are consistent with its description.

The information that structures are always finite thus is *implicit*. Although it is true it cannot be derived explicitly from our descriptions. In order to make it true we have accepted some *integrity constraints*: input descriptions have a certain form and elementary tree descriptions always contain statements of the forms $t \overset{+}{\hookleftarrow} \{t_1, \ldots, t_p\}$ and $t \overset{-}{\hookleftarrow} \{t'_1, \ldots, t'_q\}$. In fact the general processing strategy discussed above, which lets us obtain models by pairwise identification of node names is correct in virtue of another integrity constraint. It depends on the requirement that if $\mathcal{D}$ contains $t \overset{+}{\hookleftarrow} \{t_1, \ldots, t_p\}$ (or $t \overset{-}{\hookleftarrow} \{t_1, \ldots, t_p\}$), we can derive $t' \neq t''$ from $\mathcal{D}$ and the general descriptions, for all $t', t'' \in \{t_1, \ldots, t_p\}$. When writing a grammar, we must make sure that this constraint will be satisfied by new tree descriptions. But this is easy, because if $t' \neq t''$ does not follow from a given new $\mathcal{D}$ it can simply be added to it as a conjunct.

Now consider the following integrity constraint:

- Whenever an elementary tree description negatively anchors a term t it also introduces a conjunct $t' \lhd t$ for some term t'.

I.e. whenever some element is negatively anchored by an elementary descrip-

[12] In both cases t is a variable that is universally quantified over and the other terms are variables that are existentially quantified.

[13] It is a theorem about first-order logic that any theory that has arbitrarily large finite models also has an infinite model. (This follows directly from Compactness.)

tion it is also provided with a mother.[14] It is easily seen that our descriptions thus far have respected this constraint. This is good because the next statement follows from it.

- Whenever $t' \lhd^+ t$ holds in a model of one of our descriptions, there are $t_1, \ldots, t_m$ ($m \geq 0$) such that $t' \lhd t_1, t_1 \lhd t_2, \ldots, t_{m-1} \lhd t_m, t_m \lhd t$ hold in that model.

The integrity constraint thus makes sure that the relation between $\lhd^+$ and $\lhd$ is as expected. We prove the statement as follows. Suppose $t' \lhd^+ t$. Since our models are finite by the first integrity constraint, there are only a finite number of s such that $t' \lhd^+ s \lhd^+ t$. Let m be this number and assume that the theorem holds for all $m' < m$. Note that as a consequence of the way our descriptions are set up, for every non-root element there must be an explicit negative anchor. In particular, since t is not the root, there must be an elementary description $\mathcal{D}$ containing a statement $s' \dot{\leftarrow} \{s'_1, \ldots, s'_q\}$, where t is one of the $\{s'_1, \ldots, s'_q\}$. By the integrity constraint above $\mathcal{D}$ also contains a conjunct of the form $t'' \lhd t$. We consider the relation between t' and t''. It is impossible that $t' \prec t''$ by $\mathcal{A}7$, $\mathcal{A}8$ and $\mathcal{A}4$. The same axioms prevent $t'' \prec t'$ being the case, while $\mathcal{A}10$ excludes $t'' \lhd^+ t'$. By $\mathcal{A}6$ we are left with the possibilities that $t'' = t'$ or that $t' \lhd^+ t''$. In the first case we are through and conclude that $m = 0$. In the second case we can take t_m to be t'' and use induction.

The previous uses of implicit information were of a logical nature and make the basic machinery of Logical Description Grammars run smoothly. But we can easily imagine integrity constraints of a more linguistic nature. For example, up to now we have used (and we will continue to use) a mild version of X' theory to describe our trees. Again there was no explicit description to the effect that trees should satisfy the X' schema, but an integrity constraint may well be imposed on lexical descriptions so that in fact they do. The general set-up not only leaves room for this but in fact we see that many other decisions of a linguistic nature remain open as well. It should be emphasized therefore that, to a high degree, Logical Description Grammar just offers and is intended to offer a *formalism* for linguistic work, it does not in itself offer what linguists call a *theory* (although the descriptions form a theory in the logicians' sense). LDGs offer so much room for linguistic choices

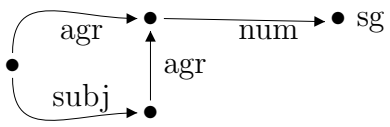
[14] The root is negatively anchored but has no mother. This means that elementary tree descriptions cannot redundantly negatively anchor the root.

of design that much of the hard linguistic work still needs to be done. The main theoretical commitment that is made by LDG is the view that linguistic representations on all levels are descriptions, not structures. This unifies the forms of representations on all levels of the grammar and, as we shall argue below, it explains the pervasiveness of ambiguity in language. But the choice for descriptions is largely orthogonal to many other linguistic choices and although we can use the notion of implicit information to implement such choices, there is nothing in the formalism that enforces them.

3.6 Adding Features

The syntactic formalism developed thus far has enough detail to illustrate our theory of syntactic and semantic underspecification, but it is too coarse-grained for actual syntactic description. For the latter we additionally need a *feature* system. We briefly consider such a feature system, basing ourselves upon the first-order axiomatisation given in (Johnson 1991).

(33)



A *feature structure* consists of a collection of feature nodes connected by labeled transitions, as in (33). We distinguish between tree nodes and feature nodes and give the latter their own type φ. The attributes labeling transitions will have type α.

(34) $\exists f\, f'\, f''[arc(f,\textsc{agr}, f') \wedge arc(f,\textsc{subj}, f'') \wedge arc(f'',\textsc{agr}, f') \wedge arc(f',\textsc{num},\textsc{sg})]$

That feature nodes are connected can be expressed using the three-place relation symbol arc,[15] with $arc(f_1, a, f_2)$ saying that f_1 and f_2 are connected by an arc labeled a. This is illustrated in (34), a statement that is satisfied by the graph in (33). Here constants such as AGR, SUBJ, NUM, PERS,... are of type α and denote attributes, while constants such as SG, PL, $1st$, $2nd$, $3rd$, $+$, $-$, ... are of type φ. We typically use them to denote graph nodes that have no successors and stand for atomic feature values. The set of all constants of type α is called C_{val}.

[15]The relation symbol arc has type $(\varphi \times \alpha \times \varphi) \rightarrow t$.

The following three axioms are a direct adaptation from (Johnson 1991). The first puts a functionality requirement on the transition relation. The second embodies the constraint that atomic features have no further attributes. And the third axiom schema, reminiscent of $\mathcal{A}11$, gives constant-constant clashes by requiring that SG $\neq$ 3rd, SG $\neq$ PL, etc.

$\mathcal{A}15$ $\forall a \forall f_1 f_2 f_3 [[arc(f_1, a, f_2) \wedge arc(f_1, a, f_3)] \rightarrow f_2 = f_3]$

$\mathcal{A}16$ $\forall a \forall f \neg\, arc(c, a, f)$, where $c \in C_{val}$

$\mathcal{A}17$ $c \neq c'$, for all syntactically distinct pairs $c, c' \in C_{val}$

We decorate tree nodes with features, writing $\boxed{n}$ for the feature decorating n (i.e. we use boxing as a function of type $\nu\varphi$). Feature information can be added to our lexical descriptions by adding conjuncts. The following example sketches how within our framework LFG-like f-structures can be combined with phrase structure trees in a way that is essentially identical to the set-up in (Kaplan and Bresnan 1982).[16] We not only need to augment the information contained in classifying descriptions, but also the information in elementary tree descriptions. The classifying description for a word like `see`, for example, could contain the information that the verb is in the present tense and requires its subject not to be third person singular:[17]

(35) $\forall k[\text{SEE}(k) \rightarrow [tv(k) \wedge \sigma(k) = saw_2 \wedge arc(\boxed{k}, \text{TENSE}, \text{PRES}) \wedge$
$\forall k'[arc(\boxed{k}, \text{SUBJ}, \boxed{k'}) \rightarrow \neg[arc(\boxed{k'}, \text{NUM}, \text{SG}) \wedge arc(\boxed{k'}, \text{PERS}, 3\text{rd})]]$

Elementary tree descriptions need also be enriched with feature information. The description for transitive verbs in (19), for example, could be made to

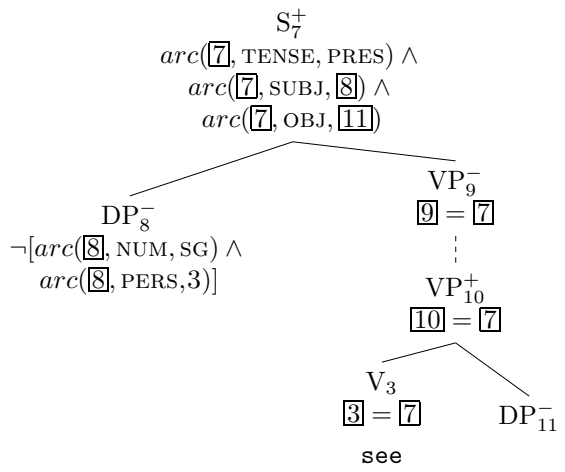
[16]The procedure sketched below can clearly also be used to provide tree nodes with feature decorations as found in the LTAGs formalism. We choose to rather eclectically combine LTAG-like phrase structures with LFG-like f-structures because a) we like to emphasize the proximity of our descriptions approach to that of LFG and stress our intellectual indebtedness to (Kaplan and Bresnan 1982) and b) we like our example to contain complex-valued features (features in LTAGs tend to be single-valued and therefore are simpler). Although we in fact do believe that the present combination of the two theories does have certain advantages over each of its components taken separately, we shall not argue for this in this paper.

[17]Of course it is a bit wasteful to give such completely regular information in each classifying description, and this can easily be avoided, but for the purpose of exposition we will leave it at this.

contain the extra conjunct in (36). This not only expresses that the feature structures of certain nodes stand in certain functional relations, but also that all nodes on the path from the lexical anchor to its maximal projection share their features. After taking witnesses and elementary reasoning, a description as in (37) is obtained.[18]

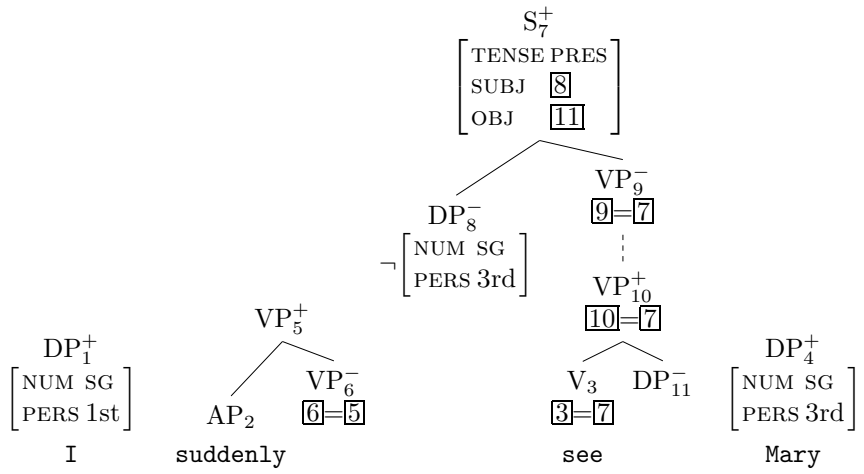
(36) $arc(\boxed{k_2}, \text{SUBJ}, \boxed{k_3}) \wedge arc(\boxed{k_2}, \text{OBJ}, \boxed{k_{11}}) \wedge \boxed{k_2} = \boxed{k_3} = \boxed{k_4} = \boxed{k_5}$

(37)



In (38) the description obtained from *I suddenly see Mary* is given, with attribute-value matrices abbreviating formulae in an obvious way.

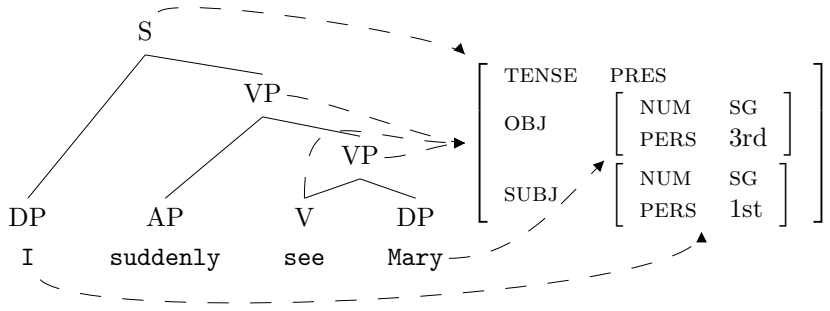
(38)



[18]We switch to graphical representations of descriptions again, but suppresss semantic information.

It is clear that this description is satisfiable. However, the extra constraints also provide a way to rule out cases where agreement is violated. Pairing off constants in (38) leads to the picture in (39), which can either be read as a description or as a minimal structure satisfying that description.[19]

(39)



It is clear that in this way we can essentially combine the phrase structure descriptions of LDG with feature structure descriptions. Again there is room for extra integrity constraints. For example, our decision that all nodes along the verbal projection line of `see` should be connected to the same feature node, could easily be made into a universal principle expressed by such an integrity constraint.[20]

3.7 Feature Structures as Semantic Representations?

Clearly, our method for describing features and the method discussed earlier for obtaining the semantics of an expression by means of codescribing its form and meaning have a lot in common. The question therefore poses itself whether the second can be reduced to the first. Can the process of assigning a semantic value to a given expression be subsumed under a general mechanism of feature assignment? There is a tradition within constraint-based linguistics which uses constraints to *describe logical sentences*. (Fenstad, Halvorsen,

[19](38) is satisfied by more than one structure, as it sets no limit on the number of feature nodes. It does restricts the number of tree nodes: there must be exactly seven of them. If (39) is interpreted as a structure, the attribute-value matrix on the right must be interpreted as a feature structure. If the picture is interpreted as a description, it is a feature description.

[20]This would in fact put restrictions on 'adjoining'. For example, if conflicting feature information were put on nodes 5 and 6 in (38), so that $\boxed{5} \neq \boxed{6}$ could be derived, no model satisfying the description would exist. Conversely, undesired adjunctions will not occur if node names that could participate in such an adjunction bear conflicting feature information.

Langholm, and van Benthem 1987), for example, propose to represent the formula $\forall x\, kick(j, x)$ as the attribute value matrix in (40). I.e. the formula is represented as a certain graph, which can then be underspecified. This possibility is used in (Nerbonne 1992) and also lies at the heart of the 'Minimal Recursion Semantics' of (Copestake, Flickinger, Malouf, Riehemann, and Sag 1995).

(40)
$$\begin{bmatrix} \text{FORMULA} & \begin{bmatrix} \text{QP} & \begin{bmatrix} \text{QUANT} & \forall \\ \text{VAR} & x \end{bmatrix} \\ \text{FORMULA} & \begin{bmatrix} \text{REL} & kick \\ \text{ARG.1} & j \\ \text{ARG.2} & x \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Although some interesting results have been obtained with the help of such descriptions of logical formulas, we think that, apart from the unwieldiness of (40) as compared with the much simpler representation $\forall x\, kick(j, x)$, there are at least two reasons to find the procedure unsatisfactory. The first objection is that linguistic representations should be used to characterise linguistic objects and that logical formulas do not qualify as such. It is one thing to assume that a logical formula is adequate for describing the meaning of a natural language expression but quite another to assume that all aspects of the particular form of such a formula are linguistically relevant. Representations like the one in (40) bring historically contingent aspects of our method to denote logical formulas into linguistic theory. The second objection is that if we represent formulas with the help of graphs and then describe graphs with the help of formulas we have something that looks very much like an epicycle. It would be much simpler to use formulas directly for describing semantic values.

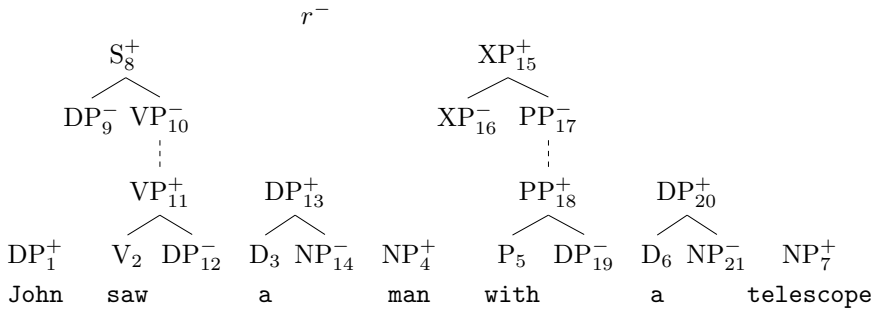
Attribute value matrices certainly provide enough structure for giving the syntax of various logics in which the semantics of natural language expressions can be expressed. But they do not seem to provide the right vehicle for doing semantics. Logical computation on such structures is certainly unwieldy (try proving $p \rightarrow p$ from a standard set of axioms in this format!) and there is no reason to assume that natural language obtains meaning via the description of some artificial language. We can use a logical language for describing the truth conditions of English, but we should not replace truth conditions with formulas.

4 Talking about Trees

The example descriptions considered in the previous section described only one tree, but this is indeed an exceptional situation. It is far more common that a multiplicity of trees satisfy a given description. Consider (42), obtained from an input description for (1a), repeated here as (41), a standard example of a PP attachment ambiguity.[21]

(41) John saw a man with a telescope.

(42)



The labeling 'XP' on 15 and 16 in (42) is a bit of a stop-gap. In fact these nodes can either be labeled VP or NP, but must of course carry the same label. This means that (43) should be part of the description obtained from the preposition.

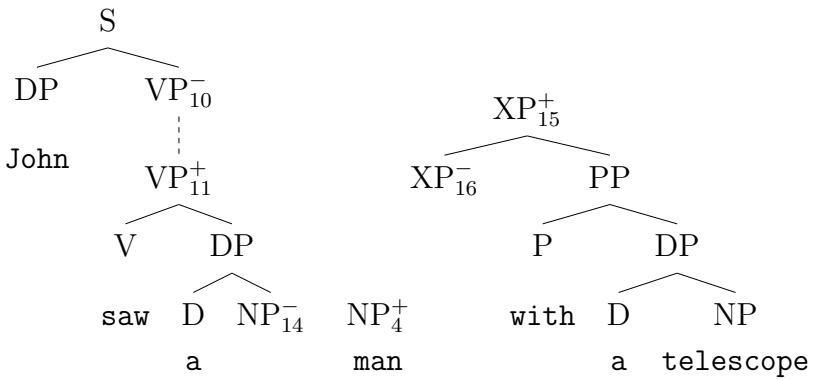
(43) $(\ell(15) = vp \vee \ell(15) = np) \wedge \ell(16) = \ell(15)$

Pairing off constants now leads to the following identifications. First we see that $21 = 7$ can be derived: 21 must match with a positively anchored name; all constants other than 4, 15 and 7 are excluded on the basis of clashing category information; but of these last the first two are out because they precede 6 while 21 is preceded by 6. In much the same way $19 = 20$, $12 = 13$, $9 = 1$, $17 = 18$ and $8 = r$ are derived. This means that we arrive at the intermediate representation in (44).[22]

[21] We suppress all semantic information in this section. The description connected with the determiner *a* is simplified for reasons of exposition. A slightly more complex form will be given shortly.

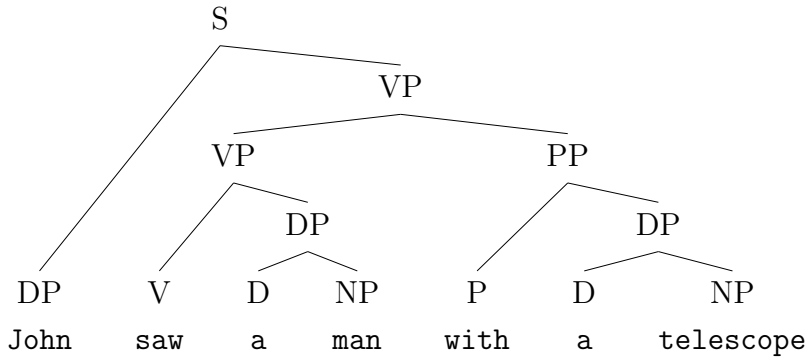
[22] Intermediate representations such as these should perhaps be compared with the 'packed shared forests' of (Tomita 1986).

(44)

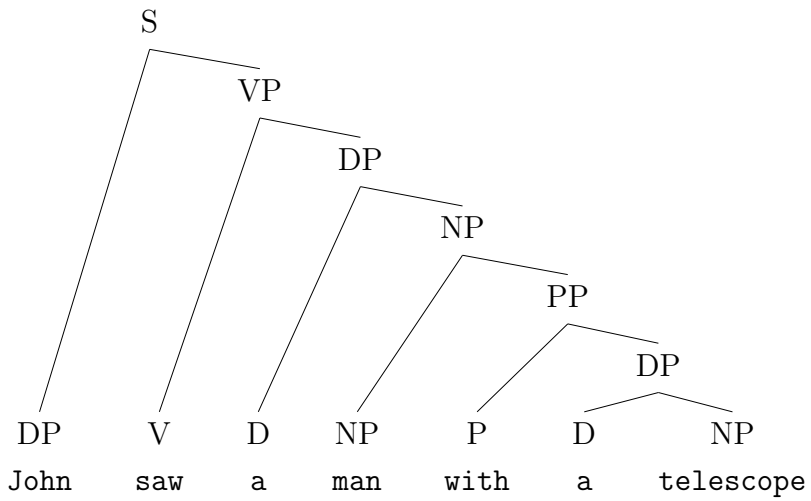


At this point there is a (first) choice. 16 can consistently be identified with 11, but also with 4. Each of these choices deterministically leads to further identifications and we end up with two models, the one in (45) and its alternative in (46).

(45)



(46)



This example shows that the descriptions approach naturally deals with structural ambiguities. In fact the input description for (41) and the representation (44) that is derived from it underdetermine the structure of (41), in the sense that more than one structure validates the description. Our second example is of a different kind. It illustrates a case where a structural approach must necessarily *over*specify the linguistic information. Consider the following three sentences, which were taken from (XTAG Research Group 1995).

(47) a. I opened up Michelle a new bank account

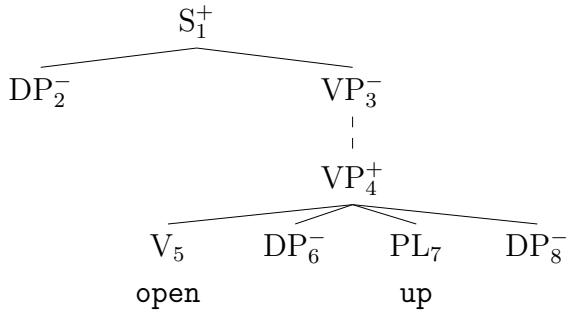
b. I opened Michelle up a new bank account

c. I opened Michelle a new bank account up

In view of this example, a structural approach must posit the existence of *three* elementary trees for the combination of words *open. . . up*. But it seems that such a treatment misses a generalisation. There is simply no relative ordering between the verb particle *up* and the two DP complements here. In a descriptions approach, on the other hand, we can easily get the required level of generality by not stipulating more relations to hold than is warranted. The description in (48) in itself would only match with an input description for (47b), but by weakening it, and putting only the constraints $5 \prec 6 \prec 8$ and $5 \prec 7$ on the relative order of the daughters of 4, we get a description which also squares with (47a) and (47c). This form of underspecification is

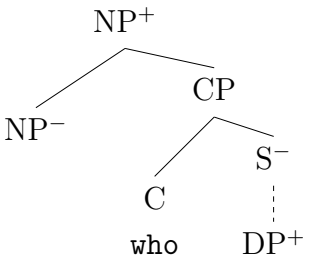
reminiscent of the Immediate Dominance / Linear Precedence (ID/LP) format of Generalised Phrase Structure Grammar (see (Gazdar, Klein, Pullum, and Sag 1985)), and indeed we feel that the ID/LP format is best viewed as a version of the descriptions approach to linguistic representation.

(48)



While the previous example illustrated the fact that left-to-right ordering can be underspecified in a descriptions approach, our third example shows that it is also possible to underpecify the distance between filler and gap in a long distance dependency.[23] Consider the following elementary tree description for the relative pronoun *who*.

(49)

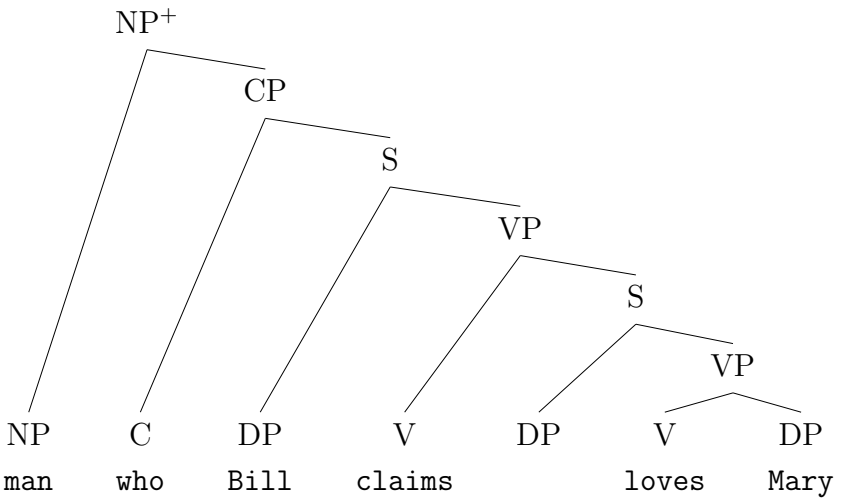


The description specifies that the relative pronoun must c-command a DP, but underspecifies the distance between the two elements. Since the DP node name is marked positively it will need to combine with a negatively marked node name lower in the tree, and the result will be the filling of a gap as in (50). In the lexical element (49) we can additionally specify dependencies between filler and gap. For example, in one dialect of English the DP in (49) should be marked for nominative, while in a similar tree for *whom* the DP should have accusative case. But there is also a semantic dependency

[23]Compare the discussion of long distance dependencies in Lexicalised Tree Adjoining Grammars given earlier.

which can be regulated at the level of the extended domain of locality for *who*. Intuitively, the semantics of *who* forms a predicate out of the semantics of its complement S and intersects this predicate with the semantics of the NP^{-} node. The DP contributes a variable (or similar) to the S semantics and the intersecting predicate is obtained by abstracting over this position. The semantics of the whole construction therefore crucially depends on that of the DP.

(50)

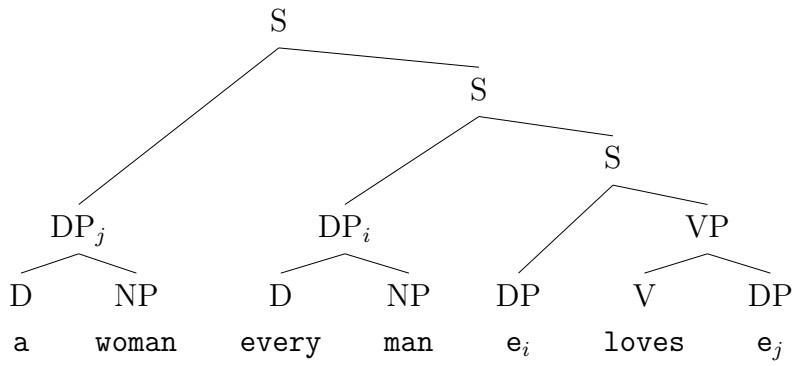


We now turn to quantifier scope ambiguities such as the one exemplified in (51). We have seen how the descriptions approach can naturally deal with structural syntactic ambiguities, due to the fact that a single description may have many structures satisfying it. Therefore, if scope ambiguities are assumed to be manifestations of structural ambiguity, they should be amenable to a similar treatment.

(51) Every man loves a woman.

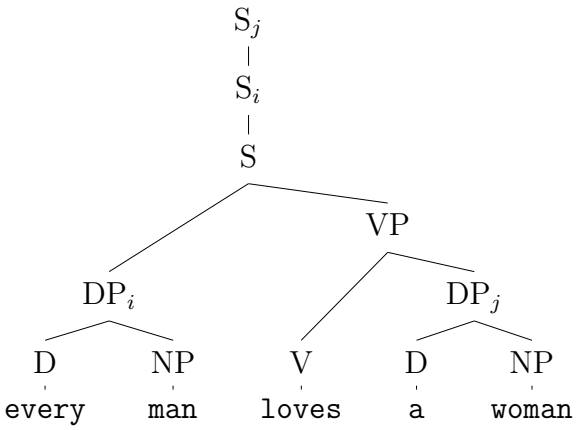
It is natural at this point to turn to (May 1977)'s Logical Forms, as these are the kind of structures required. One possible Logical Form for (51), the one that encodes the wide scope reading of the existential quantifier, would be (52). Here the DPs *every man* and *a woman* are indexed, are raised out of their surface positions, and have left a coindexed trace behind. The raising serves a semantic purpose, as it is now possible to compositionally assign the desired reading to the sentence, but a secondary result of the raising is that surface order is destroyed. This is less than desirable and in fact the movement of surface material is redundant from a semantic point of view.

(52)



In (53) we have rearranged the material in (52), leaving the DPs *in situ* syntactically, but retaining the S nodes that in May's approach result from adjoining DP to S. The result is close to the usual surface structure of the tree except that the top S node is replaced by a segment of three nodes which each will have a different semantic interpretation. The semantics of S_i will be the result of quantifying-in DP_i into S, and similarly the semantics of S_j will be the result of quantifying-in DP_j into S_i.[24]

(53)

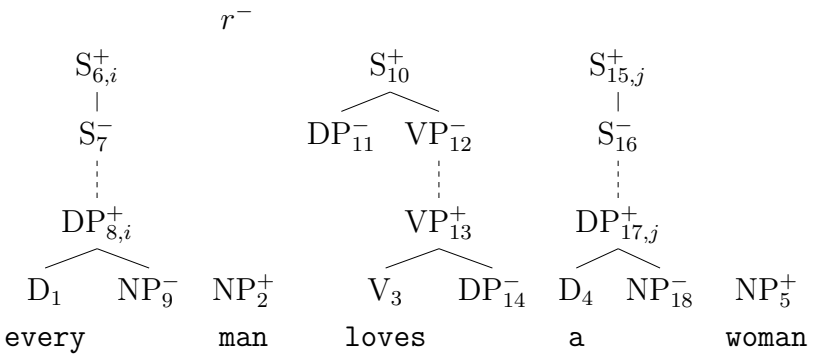


Quantifying-in is a long-distance phenomenon and long-distance phenomena are treated lexically in our theory, just as in the Tree-Adjoining Grammars of which our theory is a subspecies. Quantification will be treated within the extended domain of locality of determiners, the primary constructors

[24]The present approach may also be compared with the technique of Cooper Storage (Cooper 1983). The extra Ss then represent stages in the interpretation process of S. For example, the interpretation of S_i will be the result of retrieving the interpretation of DP_i from the store, and similarly for S_j and DP_j.

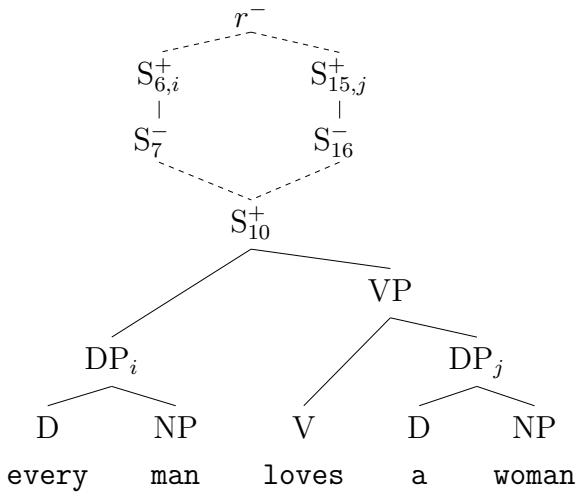
of quantifiers. In (54) the elementary tree descriptions of determiners are revised to the effect that each determiner now comes with two extra S nodes. (The coindexing between the top S and DP is just mnemonic. The indexation is not as such reflected in the underlying logical representation.)

(54)



Identifying provably corefering node names as before, we arrive at the intermediate description in (55), our 'packed' representation of the two scope possiblities.

(55)



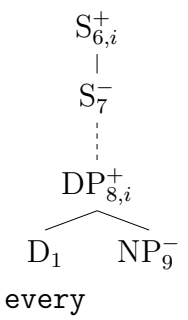
Clearly, two possibilities are left open here. Identifying 7 and 10 will lead to the possibility in (53), the tree underlying the wide scope reading of the existential quantifier. Adding $16 = 10$ to the description on the other hand will lead to the tree underlying the $\forall\exists$ reading.

5 Talking about Binding

5.1 A Difficulty with Determiners

How can we assign a semantics to the lexical descriptions in (54)? When we try to write down a semantics for the determiners, a difficulty pops up. Consider again the lexical item for *every* (repeated here in (56)). We must express the meaning of the upper S node 6 in terms of that of 7, *whatever the meaning of* 7 *turns out to be*, i.e. we must be able to express the result of quantification into an arbitrary context.[25]

(56)



It should be noted that there is a similar situation in elementary logic where the semantics of universal quantification must be explained, i.e. where the semantics of $\forall x\varphi$ must be given in terms of the semantics of φ. The way in which this is standardly solved derives from Tarski and presupposes a separation between a language described and the language describing it. This in fact fits well into our general descriptions approach which allows us to describe any given natural language with the help of logic. In order to solve our difficulty with quantification into arbitrary contexts we will emulate within the system the usual approach to binding in classical logic. First, after analyzing what is needed, we will axiomatize the binding machinery; then the correctness of our axiomatization will be shown; and finally it will be explained how the approach solves the problem with stating the semantics of (56).

[25] A naive attempt to give the required semantics would be to demand that $\sigma(6) = \forall x[\sigma(9)(x) \to \sigma(7)]$ and $\sigma(8) = x$. That this does not work is seen by considering that x is not free in the term $\sigma(7)$ and that hence, by α-conversion, $\sigma(6) = \forall y[\sigma(9)(y) \to \sigma(7)]$. Substitution of $\sigma(7)$ by a term containing a free x will not lead to the desired capturing of the variable.

5.2 Axioms for Logical Binding

Let us recap the standard Tarski definition. Suppose we have a first-order model $\mathcal{M}$, consisting of a domain $\mathcal{D}$ and an interpretation function $\mathcal{I}$. Let $\mathcal{V}$ be the set of variables in the language and define $\mathcal{A}$, the set of *assignments* for $\mathcal{M}$, as $\mathcal{D}^{\mathcal{V}}$. The statement that assignments a and b differ at most in the variable x will be written as $a[x]b$ and the value of a term t (in $\mathcal{M}$) under an assignment a, defined in the usual way, will be written as $[\![t]\!]^a$. The definition then identifies the semantic value $[\![\psi]\!]$ (in $\mathcal{M}$) of a formula ψ with a set of assignments in the following way.

$$
\begin{array}{rrcl}
(57) & [\![R(t_1,\ldots,t_n)]\!] & = & \{a \in \mathcal{A} \mid \langle [\![t_1]\!]^a,\ldots,[\![t_n]\!]^a\rangle \in \mathcal{I}(R)\} \\
& [\![t = t']\!] & = & \{a \in \mathcal{A} \mid [\![t]\!]^a = [\![t']\!]^a\} \\
& [\![\neg\varphi]\!] & = & \mathcal{A} - [\![\varphi]\!] \\
& [\![\varphi \to \psi]\!] & = & (\mathcal{A} - [\![\varphi]\!]) \cup [\![\psi]\!] \\
& [\![\forall x\varphi]\!] & = & \{a \in \mathcal{A} \mid \forall b \in \mathcal{A} : a[x]b \to b \in [\![\varphi]\!]\}
\end{array}
$$

The last clause here gives the semantics of $\forall x\varphi$ in terms of that of φ, for arbitrary φ. How can we emulate this feat *within* the classical language $\mathcal{L}$ we have been using thus far? Definition (57) is framed in mathematical English[26] and talks about predicate logic; we want to use $\mathcal{L}$ to talk about English. Mathematical English can talk about predicate logic because it has available concepts such as variables and assignments. We therefore need to have similar concepts at our disposal in $\mathcal{L}$. Of course $\mathcal{L}$ already contains variables and is interpreted in terms of assignments. But since we want to use $\mathcal{L}$ as a metalanguage for English, we also need to have similar objects at its object level.

Let us introduce such objects and call those that approximate variables *registers* (type π), and those approximating assignments *states* (type s). An essential property of the relation between variables and assignments is that, for each assignment a, each variable z and each object d of the domain under consideration, there is an assignment b such that $a[z]b$ and $b(z) = d$. The following axiom imposes that registers and states obey a similar constraint.[27]

$\mathcal{A}18\ \ \forall i_s \forall v_\pi \forall x_e\, \exists j_s\, [i[v]j \wedge V(v,j) = x]$

[26] English + set theory roughly; in (57) we mainly see set theory.

[27] The property under consideration is exactly the one which is weakened in the 'Modal State Semantics' of (Benthem 1996). This suggests that by weakening $\mathcal{A}18$ we could have some of the effects of Van Benthem's Modal State Semantics within our system. We shall not further pursue this interesting possibility here.

Here V (type $\pi \times s \rightarrow e$) is a function which assigns a value to each register v in each state j and $i[\delta]j$ is an abbreviation of $\forall v[v \neq \delta \rightarrow V(v,i) = V(v,j)]$.[28]

We need a way to generate 'fresh' registers and the following two axiom schemes will provide a mechanism for providing them. Let $\mathcal{C}$ be a set of constants of type $\nu \rightarrow \pi$. (We will use only one element u of $\mathcal{C}$ in this paper, but want the possibility of having more than one register connected wit each node) Define $\mathcal{U}$ as $\{\rho(n) \mid \rho \in \mathcal{C} \text{ and } n \in \{0,1,2,\ldots\}\}$. The elements of $\mathcal{U}$ (examples: $u(4)$, $u(23)$, often written as u_4, u_{23})) are closed terms of type π which will play the role of mimicking variables. They will be called *pseudovariables*.[29]

The functions in $\mathcal{C}$ associate every node of our trees with a number of variable registers. To make sure that these registers are really fresh, it must be prevented that certain distinct elements of $\mathcal{U}$ corefer. For example, it must be prevented that u_8 and u_{12} in fact refer to the same register. Otherwise they would encode the same variable and 'quantifying' over u_8 could possibly capture u_{12}. This noncoreference of pseudovariables is enforced by $\mathcal{A}19$ and $\mathcal{A}20$. The first of these ensures that different elements of $\mathcal{C}$ cannot assign coreferring pseudovariables to any two nodes. The second axiom requires the elements of $\mathcal{C}$ to be injective, so that they will assign different registers to different nodes.

$\mathcal{A}19$ $\forall k k' \, \rho(k) \neq \rho'(k')$, if $\rho, \rho' \in \mathcal{C}$ are syntactically distinct

$\mathcal{A}20$ $\forall k_1 k_2 \, [\rho(k_1) = \rho(k_2) \rightarrow k_1 = k_2]$, if $\rho \in \mathcal{C}$

[28]Note that for each state i there is a function $\lambda v.V(v,i)$ assigning values to registers. The converse is not necessarily true and from a formal point of view imposing $\mathcal{A}18$ is much weaker than requiring that states correspond to the full space of functions from registers to values. It might seem then that we have not captured enough properties of assignments here, but in Theorem 1 below we shall see that the essential properties are in fact captured. The reason is that our requirement that the set of assignments $\mathcal{A}$ correspond to the full function space $\mathcal{D}^{\mathcal{V}}$ is unnecessarily strong. Restricting consideration in (57) to a set $\mathcal{A}'$ of assignments differing only in finitely many places from a given assignment a, for example, would not change key notions such as satisfiability and entailment. Of course in many developments of elementary logic assignments are simply finite sequences or finite functions from variables to elements of a given domain.

[29]The idea of having analogues of variables and assignments at the object level is not new. (Sternefeld 1997) traces it back to (Bennett 1979) and in fact to (Montague 1970). (Montague did not work with an intermediary logic in EFL, but his infinite sequences of individuals A^{ω} are essentially assignments.) (Sternefeld 1997) elegantly applies a worked-out version of Bennett's theory to reconstruction and connectivity phenomena in an LF-based framework. The logical part of his theory is closely connected to ours.

This requirement on freshness of registers should carefully be distinguished from the possibility that registers get the same value in a given state. It is possible e.g. that $V(i, u_8) = V(i, u_3)$ for some i, even if $8 \neq 3$. Indeed, $\mathcal{A}18$ requires that such an i exists.

For more information on a set of axioms strongly related to $\mathcal{A}18$–$\mathcal{A}20$ see (Muskens 1996).

5.3 Embedding Predicate Logic

The axioms considered above essentially allow our logical language to talk about a form of binding. Can we capture the essentials of the binding machinery? In order to answer this question we show that it is possible to embed predicate logic into (the first-order part of) type theory, using registers and states instead of variables and assignments. Letting δ range over pseudovariables, we write

$$\begin{array}{lrll} (58) & R\{\delta_1 \ldots \delta_n\} & \text{for} & \lambda i.R(V(\delta_1, i), \ldots, V(\delta_n, i)), \\ & \delta \equiv \delta' & \text{for} & \lambda i.V(\delta, i) = V(\delta', i), \\ & -\gamma & \text{for} & \lambda i \neg \gamma(i), \\ & \gamma \supset \gamma' & \text{for} & \lambda i[\gamma(i) \rightarrow \gamma'(i)], \\ & \Pi\delta\ \gamma & \text{for} & \lambda i \forall j[i[\delta]j \rightarrow \gamma(j)]. \end{array}$$

These abbreviations mimick the Tarski truth conditions for (in that order) predication, identity, negation, implication and universal quantification. They should be compared with the clauses in (57). Here are some further abbreviations that will prove useful.

$$\begin{array}{lrll} (59) & \gamma \cap \gamma' & \text{abbreviates} & -[\gamma \supset -\gamma'] \\ & \gamma \cup \gamma' & \text{abbreviates} & -\gamma \supset \gamma' \\ & \Sigma\delta\,\gamma & \text{abbreviates} & -\Pi\delta - \gamma \end{array}$$

The abbreviations in (59) mimick conjunction, biconditional implication, disjunction and existential quantification.

In order to make precise the relation between these abbreviations and ordinary predicate logic, let us consider the language described by the following Backus-Naur Form.

$$(60)\ \gamma \quad ::= \quad R\{\delta_1, \ldots, \delta_n\} \ \mid\ \delta_1 \equiv \delta_2 \ \mid\ -\gamma \ \mid\ \gamma \supset \gamma' \ \mid\ \Pi\delta\,\gamma$$

Here the R are taken from some repository of constants (e.g. *love, walk, give,...*) of types $e \times \cdots \times e \to t$, while the δs are as before.

Although it is obvious how the fragment defined in (60) corresponds to predicate logic, we give an embedding translation for concreteness. Let $\dagger$ be a function such that $\dagger$ bijectively maps the set of pseudovariables onto the set of variables of type e. Define

$$\begin{aligned} \textsc{tr}(R\{\delta_1, \ldots, \delta_n\}) &= R(\delta_1^\dagger, \ldots, \delta_n^\dagger) \\ \textsc{tr}(\delta_1 \equiv \delta_2) &= \delta_1^\dagger = \delta_2^\dagger \\ \textsc{tr}(-\gamma) &= \neg \textsc{tr}(\gamma) \\ \textsc{tr}(\gamma \supset \gamma') &= \textsc{tr}(\gamma) \to \textsc{tr}(\gamma') \\ \textsc{tr}(\Pi\delta\, \gamma) &= \forall \delta^\dagger\, \textsc{tr}(\gamma) \end{aligned}$$

For any formula φ and state variable i, let φ^i be the result of substituting $V(\delta, i)$ for each free $\delta^\dagger$ in φ. That our fragment is really just another incarnation of predicate logic (provided that different node names refer to different nodes) is the content of the following theorem (see also (Muskens 1999)).

Theorem 1 *Let γ be a term of type $s \to t$ as defined in (60). Let i be an arbitrary variable of type s. Let Γ contain all statements $n \neq n'$ for every pair n, n' of syntactically different node names in γ. Then $\Gamma \models_{\mathcal{G}} \textsc{tr}(\gamma)^i \leftrightarrow \gamma(i)$.*

The theorem will be proved in an appendix. The requirement on node names will be met automatically in our applications.

At this point the reader may well wonder what the advantage of defining a version of predicate logic within classical type logic is. The answer is that our embedding of the variable binding mechanism a) frees us from technical difficulties with respect to substitution and b) allows us to incorporate logics in which variable binding is handled differently from the way it is handled in classical logic. In order to appreciate a) the reader may note that every term in the language generated by the Backus-Naur form in (60) is in fact *closed*. This means, for example, that the equations in (61a) and (61b) straightforwardly lead to the equation in (61c), i.e. we can substitute even while the pseudovariable u_1 intuitively 'gets bound'. On the other hand, the equation in (61c) is equivalent to the one in (61d) by the previous theorem, so in a sense we *did* manage to capture a variable.

(61) a. $\sigma(6) = \Pi u_1[man\{u_1\} \supset \sigma(7)]$

b. $\sigma(7) = dance\{u_1\}$

c. $\sigma(6) = \Pi u_1[man\{u_1\} \supset dance\{u_1\}]$

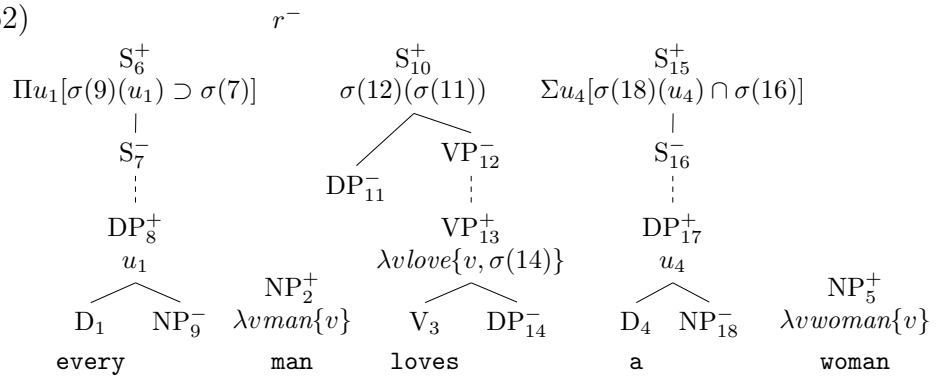
d. $\sigma(6) = \lambda i \forall x[man(x) \rightarrow dance(x)]$

The other advantage of our embedding (mentioned in b) above) is that it liberates the grammatical description of binding in natural language from an all too direct dependency on the binding mechanism in classical logic. Binding in natural languages and binding in predicate logic share many characteristics but are also different in crucial respects. The present machinery can be used to capture forms of binding other than the standard one, such as the one proposed by Kamp and Heim, as was shown in (Muskens 1996). This line will not be pursued in this paper but will be further explored in future work.

5.4 Talking about Quantification

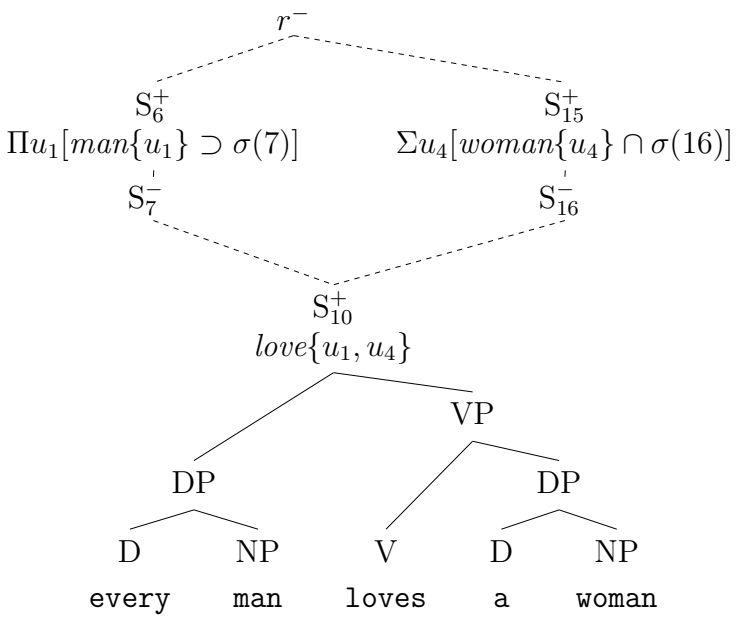
Our internalisation of the binding machinery can now be applied to quantification in natural language. In particular, we can provide the lexical items occurring in (54) with a semantics as in (62).

(62)



Reasoning as before, we arrive at a 'packed' representation as in (55). But nodes are now interpreted as in (63).

(63)



Note the similarity between the upper part of (63) and the 'Underspecified Discourse Representation Structure' in (2b). That the representation in (2b) is based upon Discourse Representation Structures while the one here is essentially based upon predicate logic is a superficial difference that will vanish if we use the embedding of the binding mechanism to arrive at an embedding of DRT within classical logic as in (Muskens 1996). The difference that is important here is that (63) is now understood as representing a phase in the reasoning process of a listener who reasons about the trees and truth-conditions that are consistent with his perceived input. The dominance statements in (63) are statements about the structure of the linguistic trees that are available and have no status independent from syntactic description as in UDRT.

It is at this point that the free substitution property of our formulas can be used. The hypothesis that $7 = 10$ immediately leads to the equation in (64a) and therefore to that in (64b), while the hypothesis that $7 \neq 10$ leads to (64c) and the equivalent (64d).

(64) a. $\sigma(r) = \Sigma u_4[woman\{u_4\} \cap \Pi u_1[man\{u_1\} \supset love\{u_1, u_4\}]]$

b. $\sigma(r) = \lambda i \exists x_4[woman(x_4) \wedge \forall x_1[man(x_1) \rightarrow love(x_1, x_4)]]$

c. $\sigma(r) = \Pi u_1[man\{u_1\} \supset \Sigma u_4[woman\{u_4\} \cap love\{u_1, u_4\}]]$

d. $\sigma(r) = \lambda i \forall x_1[man(x_1) \rightarrow \exists x_4[woman(x_4) \wedge love(x_1, x_4)]]$

It can be concluded that from the input description for *every man loves a woman* the disjunction $(64b) \vee (64d)$ can be derived.[30] But while $(64b) \vee (64d)$ is derivable from the input description it should very much be doubted if such disjunctions are in fact derived while sentences are being processed, except perhaps in a minority of cases. Since the number of readings of a sentence or text in general increases non-polynomially as a function of the length of that sentence or text, the number of disjuncts here would increase non-polynomially too. This means that we can say a priori that systematically generating such disjunctions is not feasible for the hearer.

6 Conclusion

In this paper we have shown how a simultaneous description of syntax and semantics can be given. The syntactic part of the theory is based on Tree Adjoining Grammars, broadly conceived, but uses descriptions as in the D-Tree Grammars of (Rambow, Vijay-Shanker, and Weir 1995) and adds what we have called a *chemistry of composition* that reduces the problem of parsing to the problem of identifying positively and negatively marked node names in such a way that tree constraints and category information constraints are respected. The +-nodes and −-nodes in this set-up are strongly reminiscent of positive and negative occurrences of categories in Categorial Grammar. Although the logic underlying our descriptions is classical and therefore lacks the inherent resource-sensitivity that we find in Linear Logic or the Lambek Calculus, the logic plus some extralogical axioms nevertheless model the resource-sensitivity of natural language. We conclude that a shift from classical logic to Linear Logic or the Lambek Calculus needs motivation over and above the resource-sensitivity of those systems.

[30] Note that the disjunction outscopes the two equalities here. There is *no* implication that $\sigma(r)$ is identical to any disjunction or union. A view that seems to be held very often is that the value of an ambiguous expression can be the disjunction of the values of its disambiguations. This approach seems attractive when entailments with an ambiguous sentence as premise are considered. For if $A \models B$ holds and A' is a disambiguation of A then $A' \models B$ will also hold. This is attractive as entailments should presumably be preserved under disambiguation. A problem arises however when entailments with ambiguous conclusions are considered, for if $A \models B$ holds and B' is a disjunct of B, $A \models B'$ need not hold.

In fact one thing that we hope to have shown here is that predicate logic does reasonably well as a language formalism if we are willing to axiomatise the necessary concepts. In this way it is possible to give a completely declarative account of linguistic interpretation in the sense that we can give a theory $\mathcal{G}$ consisting of general descriptions and lexical descriptions such that linguistically relevant properties of any expression (including its possible interpretations) follow from $\mathcal{G}$ plus a description $\mathcal{I}$ of the observable properties of that expression. If $\mathcal{G} + \mathcal{I}$ has more than one model then the expression described is ambiguous and the language user can try to draw in additional non-linguistic information to arrive at disambiguation, or at least to obtain a description with fewer models.

It should be emphasized that while our way to arrive at sets of structures via descriptions may be a departure from some traditional ways of generating representations, the resulting structures and the semantic values assigned to those structures are essentially of the kind that many linguists would accept. Our plea for a strictly declarative perspective and a move towards descriptions rather than structures as the prime carriers of linguistic representation has no hidden agenda. The structures that we get in the end are not significantly different from those we are used to. We see this as a virtue. The phenomenon of underspecification forces us to look for alternative ways in which linguistic information is represented, but it does not in itself force us to give up the kind of structures that were argued for on independent grounds.

Acknowledgements

This paper started as a joint project with Emiel Krahmer. That project was finally abandoned, but I would like to thank Emiel for many interesting discussions and valuable feedback. I also want to thank Kurt Eberle, Barbara Partee, Stanley Peters and all other participants of the Bad Teinach Workshop on Models of Underspecification and the Representation of Meaning (May 1998) for their comments and criticisms on an earlier version. Paul Dekker and Noor van Leusen gave detailed comments and encouragement. I thank them for both.

Finally, I wish to acknowledge a general intellectual debt to my former teacher and promotor Johan van Benthem. This debt is greater than I will ever be able to repay. A very preliminary version of this paper was on a CD given to him on his 50th birthday. The finished product is also gratefully

dedicated to Johan.

A Proof of Theorem 1

Let us write $[t/x]\varphi$ for the result of substituting t for each free x in φ. It follows from $\mathcal{A}18$ that, for each $\delta \in \mathcal{U}$,

$$\forall i\,[\forall x\,\varphi \leftrightarrow \forall j\,[i[\delta]j \rightarrow [V(\delta, j)/x]\varphi]].$$

Moreover, by $\mathcal{A}19$, $\mathcal{A}20$, the definition of $i[\delta]j$ and our assumption of non-coreference of different node names, we have that $\forall j\,[i[\delta]j \rightarrow V(\delta', j) = V(\delta', i)]$ if δ and δ' are syntactically different referents $\in \mathcal{U}$. Hence

$$\forall j\,[i[\delta]j \rightarrow [\varphi^j \leftrightarrow ([V(\delta, j)/\delta^\dagger]\varphi])^i]].$$

Using these two observations, the theorem can easily be proved with the help of an induction on the construction of γ. We do two cases here, leaving the other three to the reader. In the following '$\approx$' stands for 'is equivalent with'. We start with the case of atomic formulae.

$$\begin{aligned}
(\textsc{tr}(R\{\delta_1, \ldots, \delta_n\}))^i &\approx \\
R(\delta_1^\dagger, \ldots, \delta_n^\dagger)^i &\approx \quad \text{(by def. } (.)^i\text{)} \\
R(V(\delta_1, i), \ldots, V(\delta_n, i)) &\approx \\
R\{\delta_1, \ldots, \delta_n\}(i)
\end{aligned}$$

And here is the more interesting quantification case.

$$\begin{aligned}
\textsc{tr}(\Pi\delta\,\gamma)^i &\approx \\
(\forall\delta^\dagger\,\textsc{tr}(\gamma))^i &\approx \quad \text{(first observation)} \\
(\forall j\,[i[\delta]j \rightarrow [V(\delta, j)/\delta^\dagger]\textsc{tr}(\gamma)])^i &\approx \\
\forall j\,[i[\delta]j \rightarrow ([V(\delta, j)/\delta^\dagger]\textsc{tr}(\gamma))^i] &\approx \quad \text{(second observation)} \\
\forall j\,[i[\delta]j \rightarrow (\textsc{tr}(\gamma))^j] &\approx \quad \text{(induction)} \\
\forall j\,[i[\delta]j \rightarrow \gamma(j)] &\approx \\
\Pi\delta\,\gamma(i)
\end{aligned}$$

This ends the proof.

References

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

Bennett, M. (1979). *Questions in Montague Grammar*. Indiana University Linguistics Club.

Benthem, J. v. (1996). *Exploring Logical Dynamics*. Stanford: CSLI.

Blackburn, P. (1993). Modal Logic and Attribute-Value Structures. In M. de Rijke (Ed.), *Diamonds and Defaults*. Dordrecht: Kluwer.

Blackburn, P., C. Gardent, and W. Meyer-Viol (1993). Talking about Trees. In *Proc. of the 6th Conference of the EACL*, pp. 21–29.

Blackburn, P. and W. Meyer-Viol (1996). Modal Logic and Model-Theoretic Syntax. In M. de Rijke (Ed.), *Advances in Intensional Logic*, pp. 27–58. Dordrecht: Kluwer.

Borščev, V. and M. Xomjakov (1971). Neighbourhood Grammars and Translation: An Axiomatic Approach to the Description of Formal Languages. In *Proceedings of the 3rd International Meeting on Computational Linguistics*, Debrecen, Hungary, pp. 427–432.

Cooper, R. (1983). *Quantification and Syntactic Theory*. Dordrecht: Reidel.

Copestake, A., D. Flickinger, R. Malouf, S. Riehemann, and I. Sag (1995). Minimal Recursion Semantics. Manuscript.

Cornell, T. (1994). On Determining the Consistency of Partial Descriptions of Trees. In *Proceedings of ACL-94*.

Duchier, D. and C. Gardent (1999). A Constraint-Based Treatment of Descriptions. In H. Bunt and E. Thijsse (Eds.), *Proceedings of the 3rd International Workshop on Computational Semantics (IWCS-3)*, Tilburg, pp. 71–85.

Fenstad, J., P.-K. Halvorsen, T. Langholm, and J. van Benthem (1987). *Situations, Language and Logic*. Dordrecht: Reidel.

Gardent, C. and B. Webber (1998). Describing Discourse Semantics. In *Proceedings of the 4th TAG+ Workshop*, Philadelphia. University of Pennsylvania.

Gazdar, G., E. Klein, G. Pullum, and I. Sag (1985). *Generalized Phrase Structure Grammar*. Cambridge MA: Harvard University Press.

Johnson, M. (1991). Logic and Feature Structures. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia.

Joshi, A., L. Levy, and M. Takahashi (1975). Tree Adjunct Grammars. *Journal of the Computer and System Sciences 10*, 136–163.

Kamp, H. (1981). A Theory of Truth and Semantic Representation. In J. Groenendijk, T. Janssen, and M. Stokhof (Eds.), *Formal Methods in the Study of Language*, pp. 277–322. Amsterdam: Mathematisch Centrum.

Kamp, H. and U. Reyle (1993). *From Discourse to Logic*. Dordrecht: Kluwer.

Kaplan, R. and J. Bresnan (1982). Lexical-Functional Grammar: a Formal System for Grammatical Representation. In J. Bresnan (Ed.), *The Mental Representation of Grammatical Relations*, pp. 173–281. Cambridge, MA: The MIT Press.

Marcus, M., D. Hindle, and M. Fleck (1983). D-theory: Talking about Talking about Trees. In *Proceedings of the 21st ACL*, pp. 129–136.

May, R. (1977). *The Grammar of Quantification*. Ph. D. thesis, MIT, Cambridge.

Montague, R. (1970). English as a Formal Language. In *Formal Philosophy*. New Haven: Yale University Press.

Montague, R. (1973). The Proper Treatment of Quantification in Ordinary English. In *Formal Philosophy*, pp. 247–270. New Haven: Yale University Press.

Muskens, R. (1995). Order-Independence and Underspecification. In J. Groenendijk (Ed.), *Ellipsis, Underspecification, Events and More in Dynamic Semantics*. DYANA Deliverable R.2.2.C. (This paper also appeared in H. Kamp and B. Partee (eds.): *Context-dependence in the Analysis of Linguistic Meaning*, Stuttgart University, 1997.).

Muskens, R. (1996). Combining Montague Semantics and Discourse Representation. *Linguistics and Philosophy 19*, 143–186.

Muskens, R. (1999). Underspecified Semantics. In U. Egli and K. Von Heusinger (Eds.), *(Title unknown)*. Kluwer. To appear.

Nerbonne, J. (1992). Constraint-based Semantics. In P. Dekker and M. Stokhof (Eds.), *Proceedings of the Eighth Amsterdam Colloquium*. Amsterdam: ILLC.

Rambow, O., K. Vijay-Shanker, and D. Weir (1995). D-Tree Grammars. In *Proceedings of ACL-95*, Cambridge, MA. MIT.

Reyle, U. (1993). Dealing with Ambiguities by Underspecification: Construction, representation and deduction. *Journal of Semantics 10*, 123–179.

Rogers, J. (1996). A Model-Theoretic Framework for Theories of Syntax. In *Proc. of the 34th Annual Meeting of the ACL*.

Schabes, Y. (1990). *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph. D. thesis, University of Pennsylvania.

Sternefeld, W. (1997). The Semantics of Reconstruction and Connectivity. Technical report, Universität Tübingen. Arbeitspapiere des SFB 340.

Tomita, M. (1986). *Efficient Parsing for Natural Language*. Dordrecht: Kluwer Academic Publishers.

Vijay-Shankar, K. (1992). Using Descriptions of Trees in a Tree Adjoining Grammar. *Computational Linguistics 18*, 481–518.

Webber, B., A. Knott, and A. Joshi (1999). Multiple Discourse Connectives in a Lexicalized Grammar for Discourse. In H. Bunt and E. Thijsse (Eds.), *Proceedings of the 3rd International Workshop on Computational Semantics (IWCS-3)*, Tilburg, pp. 309–325.

XTAG Research Group (1995). A Lexicalized Tree Adjoining Grammar for English. IRCS Report 95-03, University of Pennsylvania.