Categorial Grammar and Discourse Representation Theory*

Reinhard Muskens

Abstract

In this paper it is shown how simple texts that can be parsed in a Lambek Categorial Grammar can also automatically be provided with a semantics in the form of a Discourse Representation Structure in the sense of Kamp [1981]. The assignment of meanings to texts uses the Curry-Howard-Van Benthem correspondence.

1 Introduction

In Van Benthem [1986] it was observed that the Curry-Howard correspondence between proofs and lambda terms can be used to obtain a very elegant and principled match between Lambek Categorial Grammar and Montague Semantics. Each proof in the Lambek calculus is matched with a lambda term in this approach, and Van Benthem shows how this lambda term can be interpreted as a recipe for obtaining the meaning of the expression that corresponds to the conclusion of the Lambek proof from the meanings of its constituent parts.

Usually the semantics that is obtained in this way is an extensional variant of the semantics given in Montague [1973] (Hendriks [1993] sketches how the method can be generalized for the full intensional fragment). However, it is generally acknowledged nowadays that the empirical coverage of classical Montague Grammar falls short in some important respects. Research in semantics in the last fifteen years or so has increasingly been concerned with a set of puzzles for which Montague's original system does not seem to provide

*From: *Proceedings of COLING 94*, Kyoto, Japan, 1994, pp. 508–514.

us with adequate answers. The puzzles I am referring to have to do with the intricacies of anaphoric linking. What is the mechanism behind ordinary cross-sentential anaphora, as in 'Harry has a cat. He feeds it'? Is it essentially the same mechanism as the one that is at work in the case of temporal anaphora? How is it possible that in Geach's notorious 'donkey' sentences, such as 'If a farmer owns a donkey, he beats it', the noun phrase 'a farmer' is linked to the anaphoric pronoun 'it' without its having scope over the conditional and why is it that the noun phrase is interpreted as a universal quantifier, not as an existential one?

While it has turned out rather fruitless to study these and similar questions within classical Montague Grammar (MG), they can be studied profitably within the framework of Discourse Representation Theory (DRT, see Heim [1982, 1983], Kamp [1981], Kamp & Reyle [1993]). This semantic theory offers interesting analyses of the phenomena that were mentioned above and many researchers in the field now adopt some form of DRT as the formalism underlying their semantical investigations.

But the shift of paradigm seems to have its drawbacks too. Barwise [1987] and Rooth [1987], for example, observe that the new theory does not give us the nice unified account of noun phrases as generalized quantifiers that Montague's approach had to offer and it is also clear from Kamp & Reyle [1993] that the standard DRT treatment of coordination in arbitrary categories cannot claim the elegance of the Montagovian treatment. For the purposes of this paper a third consequence of the paradigm shift is important. The Curry-Howard-Van Benthem method of providing Lambek proofs with meanings requires that meanings be expressed as typed lambda terms. Since this is not the case in standard DRT, the latter has no natural interface with Lambek Categorial Grammar.

It seems then that the niceties of MG and DRT have a complementary distribution and that considerable advantages could be gained from merging the two, provided that the best of both worlds can be retained in the merge. In fact the last eight years have witnessed a growing convergence between the two semantic frameworks. The articles by Barwise and Rooth that were mentioned above are early examples of this trend. Other important examples are Zeevat [1989] and Groenendijk & Stokhof [1990, 1991].

None of these papers gives the combination of DRT and type logic that is needed for attaching the first to Lambek's calculus, but in Muskens [forthcoming] it was shown how the necessary fusion can be obtained. The essential observation is that the meanings of DRT's discourse representation structures (boxes) are first order definable relations. They can thus be expressed within first order logic and within the first order part of ordinary type logic (i.e. the logic that was described in Church [1940], Gallin [1975] and Andrews [1986]). This allows us to treat noun phrases as expressions of a single type (a generalized kind of generalized quantifiers) and to have a simple rule for coordination in arbitrary categories (see Muskens [forthcoming] for a discussion of the latter). In this paper we build on the result and show how the system can also be attached to Lambek Categorial Grammar.

The rest of the paper consists of five main sections. The first takes us from English to Lambek proofs and the second takes us from Lambek proofs to semantical recipes. After the third section has described how we can emulate boxes in type logic, the fourth will take us from semantical recipes to boxes and the fifth from boxes to truth conditions.

2 From English to Lambek Proofs

I shall assume familiarity with Lambek's calculus and rehearse only its most elementary features. Starting with a set of *basic categories*, which for the purposes of this paper will be $\{txt, s, n, cn\}$ (for texts, sentences, names and common nouns), we define a *category* to be either a basic category or anything of one of the forms a / b or b \ a, where a and b are categories. A *sequent* is an expression $T \vdash c$, where T is a non-empty finite sequence of categories (the *antecedent*) and c (the *succedent*) is a category. A sequent is *provable* if it can be proved with the help of the following Gentzen rules.

$$\frac{}{c \vdash c}\,[AX] \qquad \frac{T, b \vdash a}{T \vdash a/b}\,[/R] \qquad \frac{T \vdash b \quad U, a, V \vdash c}{U, a/b, T, V \vdash c}\,[/L]$$

$$\frac{b, T \vdash a}{T \vdash b\backslash a}\,[\backslash R] \qquad \frac{T \vdash b \quad U, a, V \vdash c}{U, T, b\backslash a, V \vdash c}\,[\backslash L]$$

An example of a proof in this calculus is given in Fig. 1, where it is shown that $(s/(n\backslash s))/cn, cn, (n\backslash s)/n, ((s/n)\backslash s)/cn, cn \vdash s$ is a provable sequent. If the categories in the antecedent of this sequent are assigned to the words 'a', 'man', 'adores', 'a' and 'woman' respectively, we can interpret the derivability of the given sequent as saying that these words, in this order, belong to the category s.

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$cn \vdash cn$}
\AxiomC{}
\UnaryInfC{$cn \vdash cn$}
\AxiomC{}
\UnaryInfC{$n \vdash n$}
\AxiomC{}
\UnaryInfC{$n \vdash n$}
\AxiomC{}
\UnaryInfC{$s \vdash s$}
\RightLabel{$[\backslash L]$}
\BinaryInfC{$n, n\backslash n \vdash s$}
\RightLabel{$[/L]$}
\BinaryInfC{$n, (n\backslash s)/n, n \vdash s$}
\RightLabel{$[/R]$}
\UnaryInfC{$n, (n\backslash s)/n \vdash s/n$}
\AxiomC{}
\UnaryInfC{$s \vdash s$}
\RightLabel{$[\backslash L]$}
\BinaryInfC{$n, (n\backslash s)/n, (s/n)\backslash s \vdash s$}
\RightLabel{$[\backslash R]$}
\UnaryInfC{$(n\backslash s)/n, (s/n)\backslash s \vdash n\backslash s$}
\AxiomC{}
\UnaryInfC{$s \vdash s$}
\RightLabel{$[/L]$}
\BinaryInfC{$s/(n\backslash s), (n\backslash s)/n, (s/n)\backslash s \vdash s$}
\RightLabel{$[/L]$}
\BinaryInfC{$(s/(n\backslash s))/cn, cn, (n\backslash s)/n, (s/n)\backslash s \vdash s$}
\RightLabel{$[/L]$}
\BinaryInfC{$(s/(n\backslash s))/cn, cn, (n\backslash s)/n, ((s/n)\backslash s)/cn, cn \vdash s$}
\end{prooftree}
$$

Figure 1: Proof for 'a man adores a woman'

3 From Lambek Proofs to Semantical Recipes

Proof theory teaches us that there is a close correspondence between proofs and lambda terms. The lambda term which corresponds to a given proof can be obtained with the help of the so-called *Curry-Howard correspondence.* Van Benthem [1986] observed that the lambda term that we get in this way also gives us a correspondence between Lambek proofs on the one hand and the intended meanings of the resulting expressions on the other. In the present exposition of the Curry-Howard-Van Benthem correspondence I shall follow the set-up and also the notational conventions of Hendriks [1993]. For more explanation, the reader is referred to this work, to Van Benthem [1986, 1988, 1991] and to Moortgat [1988].

The idea behind the correspondence is that we match each rule in the Lambek calculus with a corresponding semantic rule and that, for each proof, we build an isomorphic tree of *semantic sequents*, which we define as expressions $T' \vdash \gamma$, where T' is a sequence of variables and γ is a lambda term with exactly the variables in T' free. The semantic rules that are to match the rules of the Lambek calculus above are as follows. (The term $\gamma[u := w(\beta)]$ is meant to denote the result of substituting $w(\beta)$ for u in γ.)

$$\frac{}{x \vdash x}[AX] \quad \frac{T', v \vdash \alpha}{T' \vdash \lambda v.\alpha}[/R] \qquad \frac{T' \vdash \beta \quad U', u, V' \vdash \gamma}{U', w, T', V' \vdash \gamma[u := w(\beta)]}[/L]$$

$$\frac{v, T' \vdash \alpha}{T' \vdash \lambda v.\alpha}[\backslash R] \qquad \frac{T' \vdash \beta \quad U', u, V' \vdash \gamma}{U', T', w, V' \vdash \gamma[u := w(\beta)]}[\backslash L]$$

Note that axioms and the rules [/L] and [\L] introduce new free variables. With respect to these some conditions hold. The first of these is that only variables that do not already occur elsewhere in the tree may be introduced. To state the second condition, we assume that some fixed function TYPE from categories to semantic types is given, such that TYPE(a / b) = TYPE(b \ a) = (TYPE(b), TYPE(a)). The condition requires that the variable x in an axiom $x \vdash x$ must be of TYPE(c) if $x \vdash x$ corresponds to $c \vdash c$ in the Lambek proof. Also, the variable w that is introduced in [/L] ([\L]) must be of (TYPE(b), TYPE(a)), where a / b (b \ a) is the active category in the corresponding sequent.

With the help of these rules we can now build a tree of semantic sequents that is isomorphic to the Lambek proof in Fig. 1; it is shown in Fig. 2. The semantic sequent at the root of this tree gives us a recipe to compute the meaning of 'a man adores a woman' once we are given the meanings of its constituting words. Let us suppose momentarily that the translation of the determiner 'a' is given as the term $\lambda P' \lambda P \exists x(P'(x) \wedge P(x))$ of type $(et)((et)t)$ and that the remaining words are translated as the terms *man, adores* and *woman* of types *et,* $e(et)$ and *et* respectively, then substituting $\lambda P' \lambda P \exists x(P'(x) \wedge P(x))$ for D and for D' in the succedent and substituting *man, adores* and *woman* for P, R and P' gives us a lambda term that readily reduces to the sentence $\exists x(man(x) \wedge \exists y(woman(y) \wedge adores(y)(x)))$.

The same recipe will assign a meaning to any sentence that consists of a determiner followed by a noun, a transitive verb, a determiner and a noun (in that order), provided that meanings for these words are given. For example, if we translate the word 'no' as $\lambda P' \lambda P \neg \exists x(P'(x) \wedge P(x))$ and 'every' as $\lambda P' \lambda P \forall x(P'(x) \rightarrow P(x))$, substitute the first term for *D,* the second for D', and *man, adores* and *woman* for P, R and P' as before, we get a term that is equivalent to $\neg \exists x(man(x) \wedge \forall y(woman(y) \rightarrow adores(y)(x)))$, the translation of 'no man adores every woman'.

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$P' \vdash P'$}
\AxiomC{}
\UnaryInfC{$P \vdash P$}
\AxiomC{}
\UnaryInfC{$v' \vdash v'$}
\AxiomC{}
\UnaryInfC{$v \vdash v$}
\AxiomC{}
\UnaryInfC{$p \vdash p$}
\RightLabel{$[\backslash L]$}
\BinaryInfC{$v, P'' \vdash P''(v)$}
\RightLabel{$[/L]$}
\BinaryInfC{$v, R, v' \vdash R(v')(v)$}
\RightLabel{$[/R]$}
\UnaryInfC{$v, R \vdash \lambda v'.R(v')(v)$}
\AxiomC{}
\UnaryInfC{$p' \vdash p'$}
\RightLabel{$[\backslash L]$}
\BinaryInfC{$v, R, Q \vdash Q(\lambda v'.R(v')(v))$}
\RightLabel{$[\backslash R]$}
\UnaryInfC{$R, Q \vdash \lambda v.Q(\lambda v'.R(v')(v))$}
\AxiomC{}
\UnaryInfC{$p'' \vdash p''$}
\RightLabel{$[/L]$}
\BinaryInfC{$Q', R, Q \vdash Q'(\lambda v.Q(\lambda v'.R(v')(v)))$}
\RightLabel{$[/L]$}
\BinaryInfC{$D, P, R, Q \vdash D(P)(\lambda v.Q(\lambda v'.R(v')(v)))$}
\RightLabel{$[/L]$}
\BinaryInfC{$D, P, R, D', P' \vdash D(P)(\lambda v.D'(P')(\lambda v'.R(v')(v)))$}
\end{prooftree}
$$

Figure 2: Semantic tree for 'a man adores a woman'

4 Boxes in Type Logic

In this section I will show that there is a natural way to emulate the DRT language in the first-order part of type logic, provided that we adopt a few axioms. This possibility to view DRT as being a fragment of ordinary type logic will enable us to define our interface between Categorial Grammar and DRT in the next section.

We shall have four types of primitive objects in our logic: apart from the ordinary cabbages and kings sort of entities (type e) and the two truth values (type t) we shall also allow for what I would like to call *pigeon-holes* or *registers* (type π) and for *states* (type s). Pigeon-holes, which are the things that are denoted by discourse referents, may be thought of as small chunks of space that can contain exactly one object (whatever its size). States may be thought of as a list of the current inhabitants of all pigeon-holes. States are very much like the program states that theoretical computer scientists talk about, which are lists of the current values of all variables in a given program at some stage of its execution.

In order to be able to impose the necessary structure on our models, we shall let V be some fixed non-logical constant of type $\pi(se)$ and denote the inhabitant of pigeon-hole u in state i with the type e term $\mathrm{V}(u)(i)$. We define $i[u_1 \ldots u_n]j$ to be short for

$$\forall v((u_1 \neq v \wedge \ldots \wedge (u_n \neq v) \rightarrow V(v)(i) = V(v)(j)) \ ,$$

a term which expresses that states i and j differ at most in $u_1,\ldots,u_n$; $i[\ ]j$ will stand for the formula $\forall v(V(v)(i) = V(v)(j))$. We impose the following axioms.

AX1 $\forall i \forall v \forall x\ \exists j(i[v]j \wedge \mathrm{V}(v)(j) = x)$
AX2 $\forall i \forall j(i[\ ]j \rightarrow i = j)$
AX3 $u \neq u'$, for each two different discourse referents (constants of type π) u and u'.

AX1 requires that for each state, each pigeon-hole and each object, there must be a second state that is just like the first one, except that the given object is an occupant of the given pigeon-hole. AX2 says that two states cannot be different if they agree in all pigeon-holes. AX3 makes sure that different discourse referents refer to different pigeon-holes, so that an update on one discourse referent will not result in a change in some other discourse referent's value.

Type logic enriched with these three first-order non-logical axioms has the very useful property that it allows us to have a form of the 'unselective binding' that seems to be omnipresent in natural language (see Lewis [1975]). Since states correspond to lists of items, quantifying over states corresponds to quantifying over such lists. The following lemma gives a precise formulation of this phenomenon; it has an elementary proof.

UNSELECTIVE BINDING LEMMA. Let $u_1,\ldots,u_n$ be constants of type π, let $x_1,\ldots,x_n$ be distinct variables of type e, let φ be a formula that does not contain j and let φ' be the result of the simultaneous substitution of $\mathrm{V}(u_1)(j)$ for x_1 and ... and $\mathrm{V}(u_n)(j)$ for x_n in φ, then:

$\models_{AX} \forall i(\exists j(i[u_1,\ldots,u_n]j \wedge \varphi') \leftrightarrow \exists x_1 \ldots \exists x_n \varphi)$
$\models_{AX} \forall i(\forall j(i[u_1,\ldots,u_n]j \rightarrow \varphi') \leftrightarrow \forall x_1 \ldots \forall x_n \varphi)$

We now come to the emulation of the DRT language in type logic. Let us fix some type s variable i and define $(u)^\dagger = \mathrm{V}(u)(i)$ for each discourse referent (constant of type π) u and $(t)^\dagger = t$ for each type e term t, and let us agree to write

$P\tau$	for	$\lambda i P(\tau)^\dagger$,
$\tau_1 R \tau_2$	for	$\lambda i(R(\tau_1)^\dagger(\tau_2)^\dagger)$,
τ_1 **is** τ_2	for	$\lambda i((\tau_1)^\dagger = (\tau_2)^\dagger)$,

if $P$is a term of type *et,* R is a term of type $e(et)$ and the τ's are either discourse referents or terms of type *e.* This gives us our basic conditions of the DRT language as terms of type *st.* In order to have complex conditions and boxes as well, we shall write

not Φ	for	$\lambda i \neg \exists j \Phi(i)(j)$,
Φ **or** Ψ	for	$\lambda i \exists j (\Phi(i)(j) \vee \Psi(i)(j))$,
$\Phi \Rightarrow \Psi$	for	$\lambda i \forall j (\Phi(i)(j) \rightarrow \exists k \Psi(j)(k))$,
$[u_1 \ldots u_n \mid \gamma_1,\ldots,\gamma_m]$	for	$\lambda i \lambda j (i[u_1,\ldots,u_n]j \wedge \gamma_1(j) \wedge \ldots \wedge \gamma_m(j))$,
$\Phi ; \Psi$	for	$\lambda i \lambda j \exists k (\Phi(i)(k) \wedge \Psi(k)(j))$.

Here Φ and Ψ stand for any term of type $s(st)$, which shall be the type we associate with boxes, and the γ's stand for conditions, terms of type *st.* $[u_1 \ldots u_n \mid \gamma_1,\ldots,\gamma_m]$ will be our linear notation for standard DRT boxes and the last clause embodies an addition to the standard DRT language: in order to be able to give compositional translations to natural language expressions and texts, we borrow the sequencing operator ';' from the usual imperative programming languages and stipulate that a sequence of boxes is again a box. The following useful lemma is easily seen to hold.

MERGING LEMMA. If $\vec{u}'$ do not occur in any of $\vec{\gamma}$ then

$$\models_{AX} [\vec{u} \mid \vec{\gamma}]; [\vec{u}' \mid \vec{\gamma}'] = [\vec{u}\vec{u}' \mid \vec{\gamma}\vec{\gamma}']$$

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

5 From Semantic Recipes to Boxes

Now that we have the DRT language as a part of type logic, connecting Lambek proofs for sentences and texts with Discourse Representation Structures is just plain sailing. All that needs to be done is to define a function TYPE of the kind described in section 3 and to specify a lexicon for some fragment of English. The general mechanism that assigns meanings to proofs will then take care of the rest. The category-to-type function TYPE is defined as follows. TYPE(txt) = TYPE(s) = s(st), TYPE(n) = π and TYPE(cn) = π(s(st)), while TYPE(a / b) = TYPE(b \ a) = (TYPE(b), TYPE(a)) in accordance with our previous requirement. It is handy to abbreviate a type of the form $\alpha_1(\ldots(\alpha_n(s(st))\ldots)$ as $[\alpha_1 \ldots \alpha_n]$, so that the type of a sentence now becomes [ ] (a box!), the type of a common noun $[\pi]$ and so on.

expr.	categories	type	translation
a_n	$(s/(n\backslash s))/cn$ $((s/n)\backslash s)/cn$	$[[\pi][\pi]]$	$\lambda P \lambda P'([u_n \mid]; P'(u_n); P(u_n))$
no_n	$(s/(n\backslash s))/cn$ $((s/n)\backslash s)/cn$	$[[\pi][\pi]]$	$\lambda P \lambda P'[\ \mid \mathbf{not}([u_n \mid]; P'(u_n); P(u_n))]$
every_n	$(s/(n\backslash s))/cn$ $((s/n)\backslash s)/cn$	$[[\pi][\pi]]$	$\lambda P \lambda P'[\ \mid ([u_n \mid]; P'(u_n)) \Rightarrow P(u_n)]$
Mary_n	$s/(n\backslash s)$ $(s/n)\backslash s$	$[[\pi]]$	$\lambda P([u_n \mid u_n \ \mathbf{is}\ \mathit{mary}]\ ;\ P(u_n))$
he_n	$s/(n\backslash s)$	$[[\pi]]$	$\lambda P(P(u_n))$
him_n	$(s/n)\backslash s$	$[[\pi]]$	$\lambda P(P(u_n))$
who	$(cn\backslash cn)/(n\backslash s)$	$[[\pi][\pi]\pi]$	$\lambda P' \lambda P \lambda v(P(v)\ ;\ P'(v))$
man	cn	$[\pi]$	$\lambda v[\ \mid \mathit{man}\ v]$
stinks	$n\backslash s$	$[\pi]$	$\lambda v[\ \mid \mathit{stinks}\ v]$
adores	$(n\backslash s)/n$	$[\pi\pi]$	$\lambda v' \lambda v[\ \mid v\ \mathit{adores}\ v']$
if	$(s/s)/s$	$[[][]]$	$\lambda pq[\ \mid p \Rightarrow q]$
.	$s\backslash(txt/s)$ $txt\backslash(txt/s)$	$[[][]]$	$\lambda pq(p\ ;\ q)$
and	$s\backslash(s/s)$	$[[][]]$	$\lambda pq(p\ ;\ q)$
or	$s\backslash(s/s)$	$[[][]]$	$\lambda pq[\ \mid p\ \mathbf{or}\ q]$

Table 1: The Lexicon

In Table 1 the lexicon for a limited fragment of English is given. The sentences in this fragment are indexed as in Barwise [1987]: possible antecedents with superscripts, anaphors with subscripts. The second column assigns one or two categories to each word in the first column, the third column lists the types that correspond to these categories according to the function TYPE and the last column gives each word a translation of this type. Here P is a variable of type $[\pi]$, p and q are variables of type [ ], and v is a variable of type π.

Let us see how this immediately provides us with a semantics. We have seen before that our Lambek analysis of (1) provides us with a semantic recipe that is reprinted as (2) below. If we substitute the translation of a[1], $\lambda P'\lambda P([u_1 \mid\,] \,;\, P'(u_1) \,;\, P(u_1))$ for D in the succedent of (2) and substitute $\lambda v[\mid \mathit{man}\ v]$ for P, we get a lambda term that after a few conversions reduces to (3). This can be reduced somewhat further, for now the merging lemma applies, and we get (4). Proceeding further in this way, we obtain (5), the desired translation of (1).

(1) A[1] man adores a[2] woman

(2) $D,P,R,D',P' \vdash D(P)(\lambda v.D'(P')(\lambda v'.R(v')(v)))$

(3) $[u_1 \mid\,] \,;\, [\mid \mathit{man}\ u_1] \,;\, D'(P')(\lambda v'.R(v')(u_1))$

(4) $[u_1 \mid \mathit{man}\ u_1] \,;\, D'(P')(\lambda v'.R(v')(u_1))$

(5) $[u_1\ u_2 \mid \mathit{man}\ u_1,\ \mathit{woman}\ u_2,\ u_1\ \mathit{adores}\ u_2]$

(6) Every[1] man adores a[2] woman

(7) $[\mid [u_1 \mid \mathit{man}\ u_1] \Rightarrow [u_2 \mid \mathit{woman}\ u_2,\ u_1\ \mathit{adores}\ u_2]]$

(8) $D,P,R,D',P' \vdash D'(P')(\lambda v'.D(P)(\lambda v.R(v')(v)))$

(9) $[u_2 \mid \mathit{woman}\ u_2,\ [u_1 \mid \mathit{man}\ u_1] \Rightarrow [\mid u_1\ \mathit{adores}\ u_2]]$

(10) A[1] man adores a[2] woman. She$_2$ abhors him$_1$

(11) $[u_1\ u_2 \mid \mathit{man}\ u_1,\ \mathit{woman}\ u_2,\ u_1\ \mathit{adores}\ u_2\ ,\ u_2\ \mathit{abhors}\ u_1]$

(12) If a[1] man bores a[2] woman she$_2$ ignores him$_1$

(13) $[\mid [u_1\ u_2 \mid \mathit{man}\ u_1,\ \mathit{woman}\ u_2,\ u_1\ \mathit{bores}\ u_2] \Rightarrow [\mid u_2\ \mathit{ignores}\ u_1]]$

The same semantical recipe can be used to obtain a translation for sentence (6), we find it in (7). But (1) and (6) have alternative derivations in the Lambek calculus too. Some of these lead to semantical recipes equivalent to (2), but others lead to recipes that are equivalent to (8) (for more explanation consult Hendriks [1993]). If we apply this recipe to the translations of the words in (6), we obtain (9), the interpretation of the sentence in which a^2 woman has a wide scope specific reading and is available for anaphoric reference from positions later in the text.

I leave it to the reader to verify that the little text in (10) translates as (11) by the same method (note that the stop separating the first and second sentences is lexicalised as an item of category $s\backslash(txt/s)$), and that (12) translates as (13). A reader who has worked himself through one or two of these examples will be happy to learn from Moortgat [1988] that there are relatively fast Prolog programs that automatically find all semantic recipes for a given sentence.

6 From Boxes to Truth Conditions

We now have a way to provide the expressions of our fragment automatically with Discourse Representation Structures which denote relations between states, but of course we are also interested in the truth conditions of a given text. These we equate with the domain of the relation that is denoted by its box translation (as is done in Groenendijk & Stokhof [1991]).

Theoretically, if we are in the possession of a box Φ, we also have its truth conditions, since these are denoted by the first-order term $\lambda i \exists j(\Phi(i)(j))$, but in practice, reducing the last term to some manageable first-order term may be a less than trivial task. Therefore we define an algorithmic function that can do the job for us. The function given will in fact be a slight extension of a similar function defined in Kamp & Reyle [1993].

First some technicalities. Define $adr(\Phi)$, the set of *active discourse referents* of a box Φ, by $adr([\vec{u} \mid \vec{\gamma}]) = \{\vec{u}\}$ and $adr(\Phi \; ; \Psi) = adr(\Phi) \cup adr(\Psi)$. Let us define $[t/u]\Gamma$, the substitution of the type e term t for the discourse referent u in the construct of the box language Γ, by letting $[t/u]u = t$ and $[t/u]u' = u'$ if $u' \neq u$; for type e terms t' we let $[t/u]t' = t'$. For complex constructs $[t/u]\Gamma$ is defined as follows.

$$
\begin{array}{lcll}
[t/u]P\tau & = & P[t/u]\tau & \\
[t/u]\tau_1 R\tau_2 & = & [t/u]\tau_1 R[t/u]\tau_2 & \\
[t/u](\tau_1 \textbf{ is } \tau_2) & = & [t/u]\tau_1 \textbf{ is } [t/u]\tau_2 & \\
[t/u]\textbf{not } \Phi & = & \textbf{not } [t/u]\Phi & \\
[t/u](\Phi \textbf{ or } \Psi) & = & [t/u]\Phi \textbf{ or } [t/u]\Psi & \\
[t/u](\Phi \Rightarrow \Psi) & = & [t/u]\Phi \Rightarrow [t/u]\Psi & \text{if } u \notin adr(\Phi) \\
[t/u](\Phi \Rightarrow \Psi) & = & [t/u]\Phi \Rightarrow \Psi & \text{if } u \in adr(\Phi) \\
[t/u][\vec{u} \mid \gamma_1, \ldots, \gamma_m] & = & [\vec{u} \mid [t/u]\gamma_1, \ldots, [t/u]\gamma_m] & \text{if } u \notin \{\vec{u}\} \\
[t/u][\vec{u} \mid \gamma_1, \ldots, \gamma_m] & = & [\vec{u} \mid \gamma_1, \ldots, \gamma_m] & \text{if } u \in \{\vec{u}\} \\
[t/u](\Phi; \Psi) & = & [t/u]\Phi; [t/u]\Psi & \text{if } u \notin adr(\Phi) \\
[t/u](\Phi; \Psi) & = & [t/u]\Phi; \Psi & \text{if } u \in adr(\Phi)
\end{array}
$$

The next definition gives our translation function † from boxes and conditions to first-order formulae. The variable x that is appearing in the sixth and eighth clauses is supposed to be *fresh* in both cases, i.e. it is defined to be the first variable in some fixed ordering that does not occur (at all) in Φ or in Ψ. Note that the sequencing operation ; is associative: $\Phi; (\Psi; \Xi)$ is equivalent with $(\Phi; \Psi); \Xi$ for all Φ, Ψ and Ξ. This means that we may assume that all boxes are either of the form $[\vec{u} \mid \vec{\gamma}]; \Phi$ or of the form $[\vec{u} \mid \vec{\gamma}]$. We shall use the form $[\vec{u} \mid \vec{\gamma}]; \Phi$ to cover both cases, thus allowing the possibility that Φ is empty. If Φ is empty, $\Phi \Rightarrow \Psi$ denotes Ψ.

$$
\begin{array}{lcl}
(P\tau)^\dagger & = & P(\tau)^\dagger \\
(\tau_1 R\tau_2)^\dagger & = & R(\tau_1)^\dagger(\tau_2)^\dagger \\
(\tau_1 \textbf{ is } \tau_2)^\dagger & = & (\tau_1)^\dagger = (\tau_2)^\dagger \\
(\textbf{not } \Phi)^\dagger & = & \neg(\Phi)^\dagger \\
(\Phi \textbf{ or } \Psi)^\dagger & = & \Phi^\dagger \vee \Psi^\dagger \\
(([u\vec{u} \mid \vec{\gamma}]; \Phi) \Rightarrow \Psi)^\dagger & = & \forall x([x/u](([\vec{u} \mid \vec{\gamma}]; \Phi) \Rightarrow \Psi))^\dagger \\
(([\ \mid \gamma_1, \ldots, \gamma_m]; \Phi) \Rightarrow \Psi)^\dagger & = & (\gamma_1^\dagger \wedge \ldots \wedge \gamma_m^\dagger) \rightarrow (\Phi \Rightarrow \Psi)^\dagger \\
([u\vec{u} \mid \vec{\gamma}]; \Phi)^\dagger & = & \exists x([x/u]([\vec{u} \mid \vec{\gamma}]; \Phi))^\dagger \\
([\ \mid \gamma_1, \ldots, \gamma_m]; \Phi)^\dagger & = & \gamma_1^\dagger \wedge \ldots \wedge \gamma_m^\dagger \wedge \Phi^\dagger
\end{array}
$$

By way of example, the reader may verify that the function † sends (14) to (15).

(14) $[\ \mid [u_1\ u_2 \mid man\ u_1,\ woman\ u_2,\ u_1\ bores\ u_2] \Rightarrow [\ \mid u_2\ ignores\ u_1]]$

(15) $\forall x_1 x_2((man(x_1) \wedge woman(x_2) \wedge bores(x_1)(x_2)) \rightarrow ignores(x_2)(x_1))$

It is clear that the function † is algorithmic: at each stage in the reduction of a box or condition it is determined what step should be taken. The following theorem, which has a surprisingly tedious proof, says that the function does what it is intended to do.

THEOREM. For all conditions γ and boxes Φ:

$$\begin{array}{l} \models_{AX} \lambda i \Phi^{\dagger} = \lambda i \exists j (\Phi(i)(j)) \\ \models_{AX} \lambda i \gamma^{\dagger} = \gamma \end{array}$$

References

[1] Andrews, P.B.: 1986, *An Introduction to Mathematical Logic and Type Theory: to Truth through Proof,* Academic Press, Orlando, Florida.

[2] Barwise, J.: 1987, Noun Phrases, Generalized Quantifiers and Anaphora, in P. Grdenfors (ed.), *Generalized Quantifiers,* Reidel, Dordrecht, 1-29.

[3] Van Benthem, J.F.A.K.: 1986, *Essays in Logical Semantics,* Reidel, Dordrecht.

[4] Van Benthem, J.F.A.K.: 1988, The Lambek Calculus, in: R.E. Oehrle, E. Bach and D. Wheeler (eds.), 1988, *Categorial Grammars and Natural Language Structures,* Reidel, Dordrecht.

[5] Van Benthem, J.F.A.K.: 1991, *Language in Action,* North-Holland, Amsterdam.

[6] Church, A.: 1940, A Formulation of the Simple Theory of Types, *The Journal of Symbolic Logic* **5**, 56-68.

[7] Gallin, D.: 1975, *Intensional and Higher-Order Modal Logic*, North-Holland, Amsterdam.

[8] Groenendijk, J. and Stokhof, M.: 1990, Dynamic Montague Grammar, in L. Klmn and L. Plos (eds.), *Papers from the Second Symposium on Logic and Language,* Akadmiai Kiad, Budapest, 3-48.

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

[10] Heim, I.: 1982, *The Semantics of Definite and Indefinite Noun Phrases,* Dissertation, University of Massachusetts, Amherst. Published in 1989 by Garland, New York.

[11] Heim, I.: 1983, File Change Semantics and the Familiarity Theory of Definiteness, in R. Buerle, Ch. Schwarze and A. von Stechow (eds.), *Meaning, Use and Interpretation of Language,* De Gruyter, Berlin, 164-189.

[12] Hendriks, H.: 1993, *Studied Flexibility*, ILLC Dissertation Series, ILLC, University of Amsterdam.

[13] Janssen, T.: 1983, *Foundations and Applications of Montague Grammar,* Dissertation, University of Amsterdam. Published in 1986 by CWI, Amsterdam.

[14] Kamp, H.: 1981, A Theory of Truth and Semantic Representation, in J. Groenendijk, Th. Janssen, and M. Stokhof (eds.), *Formal Methods in the Study of Language, Part I,* Mathematisch Centrum, Amsterdam, 277-322.

[15] Kamp, H. and Reyle, U.: 1993, *From Discourse to Logic,* Kluwer, Dordrecht.

[16] Lewis, D.: 1975, Adverbs of Quantification, in E. Keenan (ed.), *Formal Semantics of Natural Language,* Cambridge University Press, 3-15.

[17] Montague, R.: 1973, The Proper Treatment of Quantification in Ordinary English, in R. Montague, *Formal Philosophy*, Yale University Press, New Haven, 1974, 247-270.

[18] Moortgat, M.: 1988, *Categorial Investigations,* Foris, Dordrecht.

[19] Muskens, R.A.: forthcoming, A Compositional Discourse Representation Theory, to appear in the proceedings of the Ninth Amsterdam Colloquium.

[20] Rooth, M.: 1987, *Noun Phrase Interpretation in Montague Grammar, File Change Semantics, and Situation Semantics,* in P. Grdenfors (ed.), *Generalized Quantifiers,* Reidel, Dordrecht, 237-268.

[21] Zeevat, H.: 1989, A Compositional Approach to Discourse Representation Theory, *Linguistics and Philosophy* **12**, 95-131.