From: Urs Egli, Peter E. Pause, Christoph Schwarze, Arnim von Stechow, and Götz Wienold (eds.), *Lexical Knowledge in the Organization of Language,* Benjamins, Amsterdam, 1995, pp. 147-183.

Tense and the Logic of Change

Reinhard Muskens

INTRODUCTION

In this paper I shall show that the DRT (Discourse Representation Theory) treatment of temporal anaphora[1] can be formalized within a version of Montague Semantics that is based on classical type logic. This emulation has at least two purposes. In the first place it may serve as one more illustration of the general point that although there are several different frameworks for semantical analysis in the market today, each with its own special rhetoric and technical set-up, we often find on closer examination that these approaches are much less different than some of their proponents want us to believe.[2] The frameworks that are under consideration here, DRT and Montague Grammar, may both profit from knowing where exactly they differ and where they agree and in this paper it is shown that they do not necessarily differ in their treatment of temporal phenomena. Our reformulation also shows that we may be able to get rid of the level of discourse representations that is characteristic of DRT. Since we can express large parts of DRT in Montague Grammar and since Montague Grammar makes no essential use of a level that is intermediate between syntax and interpretation, we may conclude that we are not forced to adopt such an extra level. It is possible to make the same predictions while assuming less entities. If nothing compels us to assume the existence of representations, we should apply Occam's razor to trim them away.

The second purpose of our reformalization is to extend the DRT analysis of tense to the subclausal level. In the DRT approach whole clauses are taken as atomic while in our set-up it will be possible to study how the meaning of an expression is built from the meanings of its parts, how the meanings of those parts are built from the meanings of other expressions, and so on, down to the level of single words. Languages can build an infinity of meanings from a finite stock and it seems that we can account for this only by accepting some building block theory of meaning.[3] A theory of tense should describe what the temporal operators contribute to the meaning of the sentences and texts they occur in.

---

1 As exemplified in Kamp [1979], Hinrichs [1981, 1986], Kamp & Rohrer [1983] and Partee [1984].

2 For another illustration of this point see Muskens [1989a], or better, Muskens [1989b], in which I emulate an early version of Situation Semantics within Montague Grammar.

3 This in itself does not entail that we must embrace the strict compositionality that is the norm in Montague Grammar, of course.

This is not the first paper that fuses DRT and Montague Semantics. For example Rooth [1987], who bases himself upon Barwise [1987], gives a Montagovian reanalysis of the DRT treatment of nominal anaphora and Groenendijk & Stokhof [1990] for the same purpose develop a system called Dynamic Montague Grammar (DMG), a generalization of their Dynamic Predicate Logic (DPL; Groenendijk & Stokhof [1991]). But in this paper I'll use a system of dynamic semantics that I have formulated in Muskens [1991], a theory that, even though it borrows many ideas from DPL, uses *classical* logic only.[4] Standard DMG is based on a highly complicated logic, and it is here too that I want to apply Occam's razor. Logics ought not to be multiplied except from necessity. In order to keep things as simple as I can, I shall not make any use of the devilish confetti of boxes, cups, caps and tense operators that we find in Montague's **IL** and I shall also refrain from using the 'state switchers', the 'ups' and 'downs', the special 'quantifiers' (that are no quantifiers in the usual sense) etc. that are to be found in DMG. All these are redundant and we can stick to ordinary higher order logic (with lambdas). The structure that is needed to get a dynamic system can be obtained by using axioms. There is a price to be paid though: in general our formulae will be relatively long. Since we restrict ourselves to classical type theory our formulae will have to display all information that in specialized logics can be encoded in the system itself. On the whole, however, I think that the much greater transparency of classical logic in practice far outweighs any abbreviatory advantages that can be obtained by complicating the logic.[5]

The organization of the paper is as follows. In the next section I'll sketch a general picture of the dynamic interpretation of discourse. According to this picture a text acts upon its reader much in the same way as a computer program acts upon the machine that it runs on, bringing him from one state to another and changing the values of certain variables he is keeping track of. The axioms that I have just referred to model this change and are given in section 3. The resulting system provides us with the tools that we need for our purposes: on the one hand the axioms enable us to deal with dynamic phenomena and to take a computational approach to natural language semantics, while on the other the availability of lambdas allows us to build up meanings compositionally in the usual Montagovian manner. That Discourse Representation Theory can really be emulated in classical logic in this way is shown in section 4 where

---

4 I work in a many-sorted version of the logic in Church [1940] here. For the two-sorted variant see Gallin [1975]; a generalization to a type logic with an arbitrary finite number of basic types is trivial.

5 I think this practical reason to prefer ordinary logic is much more important than any theoretical consideration. Logicians should note however that **IL** does not have the diamond property: Friedman & Warren [1980] give the following example. Consider the term $\lambda x(\lambda y({}^\wedge y = f(x))x)c$, where x and y are variables of some type α, c is a constant of that type and f is a variable of type $(\alpha(s\alpha))$. Two reductions are possible. We can reduce to the term $\lambda y({}^\wedge y = f(c))c$ which cannot be reduced any futher, but we can also reduce the inner redex in order to obtain $\lambda x({}^\wedge x = f(x))c$. This term neither can be simplified any further. This means that in **IL** a choice which simplification to make first may crucially matter for the result.

this theory is embedded into our system, both in a direct way and via an embedding into a version of the Quantificational Dynamic Logic that is used in the study of computation. Quantificational Dynamic Logic too can be embedded into our type logic enriched with axioms.

These embeddings are not given for their technical interest primarily, but for the light that they shed on our subsequent treatment of two fragments of English. In section 5 I give the first of these. It contains nominal anaphora, and the section is in fact a quick rehearsal of the theory that was given in Muskens [1991]. For our second fragment, however, we need some more structure and in section 6 a basic ontology of eventualities and periods of time is developed. At that point the ground will be prepared for our treatment of temporal anaphora in section 7. The treatment will combine insights from Reichenbach [1947], the DRT tradition and Montague Grammar.

2. CHANGING THE CONTEXT

The reader of a text must keep track of a list of items. While he is reading, the values of these items may shift. For example, in the first sentence of the short dialogue in (1) below (an exchange between a Cop and a Robber) the *reference time* shifts several times, so that the Robber's purchase of the gun, his walking to the bank and his entering the bank are interpreted as occurring in succession. At the turn of the dialogue, the *speaker* becomes the *addressee* and the *addressee* becomes *speaker,* so that the words 'I' and 'you' are interpreted correctly. Moreover, the two indefinite noun phrases in the first sentence each create a *discourse referent* that can be picked up at later times by pronouns or definites. (Anaphoric linkage is represented by coindexing here.)

(1) —You bought a_1 gun, walked to a_2 bank and entered it_2.
i *j* *k*

—But I didn't use the_1 gun or rob the_2 bank.
l

At each point in the text the reader may be thought to be in a certain contextual *state;* [6] in each contextual state items like *reference time, speaker, addressee,* various discourse referents and so on have certain values. We may thus identify these context items with functions that take states as their arguments and assign values of an appropriate kind to them.

i *j* *k* *l*

[6] Compare the 'conversational score' in Lewis [1979].

R:	t_1	t_2	t_3	t_4
speaker:	*Cop*	*Cop*	*Cop*	*Robber*
addressee:	*Robber*	*Robber*	*Robber*	*Cop*
v_1:	?	*S&W*	*S&W*	*S&W*
v_2:	?	?	*ABN*	*ABN*

figure 1

Suppose that our reader is in some state i when he starts to read dialogue (1). Suppose also that at this stage some initial reference time t_1 is given and that it is settled that speaker and addressee are the Cop and the Robber respectively. Now reading a small portion of the text causes the reader's list of items to change. Just after the word *gun* has been processed, the reference time R has moved forward and a discourse referent for *a gun* has been stored. So now the reader is in a state j that differs from i in the two respects that R has shifted to some time[7] t_2 just after t_1 and that some gun (a Smith & Wesson, say) is stored in a discourse marker v_1.[8] A bit later, when the word *bank* is read, the reference time has moved to a period of time t_3 just after t_2 and some bank (say the ABN bank at the Vijzelgracht in Amsterdam) is now the value of a discourse marker v_2. So at this point the reader may be in a state k such that $R(k) = t_3$ and $v_2(k) = ABN$. The other items remain unchanged, so that $speaker(k) = speaker(j) = Cop$, $v_1(k) = v_1(j) = S\&W$ and $addressee(k) = addressee(j) = Robber$.

At the start of the second sentence the values for *speaker* and *addressee* are swapped, so that our reader is now in a state l such that $speaker(l) = Robber$ and $addressee(l) = Cop$. The interpretation of the VP *entered it* has caused the reference time to move again, so that $R(l) = t_4$ for some stretch of time t_4 just after t_3 ; the other items remain unaltered.

Note that we must view the interpretation process as highly *non-deterministic* if we want to model it in this way. If we allow the reader only a single list of items we must allow him many choices as to which objects he is going to store as the values for these items. For example, our reader has chosen some particular Smith & Wesson to store as the value for the discourse referent connected with *a gun,* but of course he might have taken any gun he liked; he might have chosen another value for the referent of *a bank* as well. So a reader who starts reading (1) being in state i does not *need* to end up in state l at the turn of the dialogue, although he *may* be in that state then. The state that the reader is in at a given point in the text does not depend functionally on preceding states. Given an input state, the processing of a text may lead to many different output states. We identify the meaning of a text with the binary relation consisting of all tuples

---

7 In section 7 below the point of reference R will range over eventualities.

8 The value of v_1 in state i is unimportant. We may either assume that v_1 is undefined for state i, or that it assumes some special dummy value, or that its value is completely arbitrary. Technically we take the last option.

$\langle i, j \rangle$ such that starting from state i after interpreting the text the reader may be in state j.

There is an analogy between texts and computer programs here that should be pointed out. Just like the evaluation of a text causes the values in a list of contextual items to change, running a program causes the values that are assigned to the program's variables to be altered. Our contextual states correspond to program states, and our contextual items correspond to the variables in a program. Although it is true that programs on an actual computer are deterministic—it is always completely decided what action will come next—program semanticists have found it useful to consider nondeterministic programs as well. These nondeterministic programs allow the machine to choose which of two actions it wants to perform, or to choose how many times it wants to iterate a given action.

3. THE LOGIC OF CHANGE

Let's formalize our talk about state change within the Theory of Types. We can assume that states are primitive objects (type s), and that our contextual items are functions that take states as arguments. Consequently, contextual items (or, *stores*) have a type $s\alpha$, where α is the type of the values to be stored. Let's agree to have only a finite set of types α that the values of stores can have and let's call this set Θ. In the sections to follow, Θ will be the four element set $\{e, \tau, \varepsilon, w\}$, where e is the type of individuals, τ is the type of periods of time, ε is the type of eventualities and w is the type of possible worlds, so that stores can only contain individuals, periods of time, eventualities or worlds. But for the sake of generality I'll formulate the basic theory for arbitrary finite Θ here.

Of course state changes are highly controlled and a short piece of text can only alter the values of a few stores. We need a simple way to express the fact that two states agree up to the values of some given stores. In particular, we want to have at our disposal a formula $i[v]j$ that says that all stores except v return the same value for arguments i and j. We can get such a formula in the following way. Introduce for each $\alpha \in \Theta$ a constant $ST_{(s\alpha)t}$ of type $(s\alpha)t$. This constant intuitively stands for the predicate "is a store of type $s\alpha$". Now suppose that v is a term of type $s\alpha$, then we want $i[v]j$ to mean that (a) states i and j agree in all stores of type $s\alpha$, except possibly in v and that (b) i and j agree in all stores of all other types. Here is a definition that ensures this.

DEFINITION 1. If v is a term of type $s\alpha$ ($\alpha \in \Theta$), then $i[v]j$ abbreviates the conjunction of

(a) $\quad \forall u_{s\alpha}((ST_{(s\alpha)t}u \wedge u \neq v) \rightarrow uj = ui)$ $\quad$ and

(b) the conjunction of all $\forall u_{s\beta}(ST_{(s\beta)t}\, u \rightarrow uj = ui)$ for all $\beta \in \Theta - \{\alpha\}$.

The denotation of $\lambda ij\, i[v]j$ is of course an equivalence relation.

There is an important constraint to be imposed. We want only models in which each state can be changed ad lib. Until now there is nothing that guarantees this. For example, some of our typed models may have only one state in their domain D_s of states. In models that do not have enough states an attempt to update a store may fail; we want to rule out such models. In fact, we want to make sure that we can *always* update a store selectively with each appropriate value we may like to. This we can do by means of the following axiom scheme.[9]

AX1 $\forall i \forall v_{s\alpha} \forall x_{\alpha}(ST_{(s\alpha)t}\, v \rightarrow \exists j\,(i[v]j \wedge vj = x))$ for all $\alpha \in \Theta$

The axiom scheme is closely connected to Goldblatt's [1987, pp. 102] requirement of 'Having Enough States' and with Janssen's 'Update Postulate'. We'll refer to it as the Update Axiom (Scheme). It follows from the axiom that not all type $s\alpha$ functions are stores (except in the marginal case that D_{α} contains only one element), since, for example, a constant function that assigns the same value to *all* states cannot be updated to another value. The Update Axiom imposes the condition that contents of stores may be varied at will.

Below we shall use non-logical constants (R for the point of reference, S for speech time, W for the current world, $v_0, v_1, v_2, \ldots$ for various discourse markers etc.) that are meant to refer to stores. We call these special constants *store names* and we ensure that store names refer to stores (and hence can be updated) by a simple axiom scheme.

AX2 $ST_{(s\alpha)t}\, v$ for each store name v of type $s\alpha$, for each $\alpha \in \Theta$

Although two different stores may of course have the same value at a given state, we don't want two different store names to refer to the same store. From $i[v]j$ we want to be able to conclude that $ui = uj$ if the store names u and v are syntactically different. We enforce this by demanding that

[9] Warning: some choices for Θ lead to inconsistency. For example, if we choose st—sets of states—to be an element of Θ we're in trouble. First, we get a problem with the cardinality of the set of all states, since AX1 now would require that there are at least as many states as there are sets of states, which is impossible. This problem could be circumvented by a suitable weakening of AX1: instead of requiring that stores can be updated with *any* existent set we may require that stores can be updated with any set that is the value of some closed term in the language. In applications this would be quite sufficient. The second problem is somewhat trickier. Suppose V is a type $s(st)$ store name. Consider the term $\lambda i \neg Vii$. By AX1 there is some j such that $Vj = \lambda i \neg Vii$. This immediately gives $Vjj = \neg Vjj$, a contradiction. This diagonal argument is of course strongly reminiscent of the reasoning used in the Liar paradox and in Russell's paradox. In this paper I evade the problem by simply not letting any element of Θ contain an s. In a future paper I intend to tackle the problem by choosing partial type theory as the underlying logic.

AX3 $u \neq v$ for each two different store names u and v of any type $s\alpha$.

This is our logic of change: classical type theory plus our definition for $i\,[v\,]j$ plus the three axiom schemes given above. It is useful to extend the definition of $i\,[v\,]j$ to an arbitrary finite number of stores: by induction define $i\,[u_1, \ldots, u_n\,]j$ to abbreviate the formula $\exists k\,(i\,[u_1\,]k \wedge k\,[u_2, \ldots, u_n\,]j\,)$. Given the Update Axiom we can informally paraphrase $i\,[u_1, \ldots, u_n\,]j$ as: 'states i and j agree in all stores except possibly in $u_1, \ldots, u_n$'.

The following fact is very useful. I call it the *Unselective Binding Lemma* and it has an elementary proof. Since a state can be thought of as a list of items, a quantifier over states can have the effect of a series of quantifications.

UNSELECTIVE BINDING LEMMA.[10] Let $u_1, \ldots, u_n$ be store names, not necessarily of one and the same type, let $x_1, \ldots, x_n$ be distinct variables, such that x_k is of type α if u_k is of type $s\alpha$, let φ be a formula that does not contain j and let $[u_1j\,/\,x_1, \ldots, u_nj\,/\,x_n\,]\varphi$ stand for the simultaneous substitution of u_1j for x_1 and . . . and u_nj for x_n in φ, then:

(i) AX1-AX3 $\models \exists j\;\,(i\,[u_1, \ldots, u_n\,]j \;\wedge [u_1j\,/\,x_1, \ldots, u_nj\,/\,x_n\,]\varphi\,) \leftrightarrow \exists x_1 \ldots x_n\,\varphi$

(ii) AX1-AX3 $\models \forall j\;\,(i\,[u_1, \ldots, u_n\,]j \;\rightarrow [u_1j\,/\,x_1, \ldots, u_nj\,/\,x_n\,]\varphi\,) \leftrightarrow \forall x_1 \ldots x_n\,\varphi$

4. DISCOURSE REPRESENTATION THEORY, DYNAMIC LOGIC AND TYPE THEORY

That the preceding exercise has some relevance for the dynamic interpretation of natural language may be more easily appreciated if we consider the relation between our logic and Discourse Representation Theory. I'll assume familiarity with DRT here, rehearsing only the basic facts. Definition 2 below characterizes the DRT language. The expressions in this language can be divided into two categories: conditions and Discourse Representation Structures (DRSs). As we see from the first clause, the atomic formulae of ordinary predicate logic (without function symbols) are conditions. The other two clauses allow us to build complex conditions and DRSs from simpler constructs.

---

[10] Another form of the Lemma states the equivalence of $\exists j\;\,[u_1j\,/\,x_1, \ldots, u_nj\,/\,x_n\,]\varphi$ and $\exists x_1 \ldots x_n\,\varphi$ if φ does not contain j.

DEFINITION 2 (DRT Syntax).[11]

i. If R is an n- ary relation constant and $t_1,...,t_n$ are terms (constants or variables), then $Rt_1...t_n$ is a condition.
If t_1 and t_2 are terms then $t_1 = t_2$ is a condition.

ii. If Φ and Ψ are DRSs then $\neg\Phi$, $\Phi \vee \Psi$ and $\Phi \Rightarrow \Psi$ are conditions.

iii. If $\varphi_1,...,\varphi_m$ are conditions ($m \geq 1$) and $x_1,...,x_n$ are variables ($n \geq 0$), then $[x_1 ... x_n][\varphi_1 ,..., \varphi_m]$ is a DRS.

The language is evaluated on first-order models in the following way. Let $M = \langle D, I \rangle$ be such a model (where D is the domain and I is the interpretation function of M). The value of a term t in M under an assignment a, written as $\|t\|^{M,a}$, is defined as $I(t)$ if t is a constant and as $a(t)$ if t is a variable. Definition 3 assigns a value $\|\delta\|^M$ in M to each condition or DRS δ; the value of a condition will be a set of assignments, the value of a DRS a binary relation between assignments for M. (In the definition I suppress all superscripts M and write $a\,[x_1...x_n]\,a'$ to mean that the assignments a and a' return the same values for all variables, except possibly for $x_1,...,x_n$).

DEFINITION 3 (DRT Semantics).

i. $\|Rt_1 ... t_n\| = \{a \mid \langle \|t_1\|^a, ... , \|t_n\|^a \rangle \in I(R)\}$
$\|t_1 = t_2\| = \{a \mid \langle \|t_1\|^a = \|t_2\|^a \rangle\}$

ii. $\|\neg\Phi\| = \{a \mid \neg\exists a' \langle a, a' \rangle \in \|\Phi\|\}$
$\|\Phi \vee \Psi\| = \{a \mid \exists a' (\langle a, a' \rangle \in \|\Phi\| \vee \langle a, a' \rangle \in \|\Psi\|)\}$
$\|\Phi \Rightarrow \Psi\| = \{a \mid \forall a' (\langle a, a' \rangle \in \|\Phi\| \rightarrow \exists a'' \langle a', a'' \rangle \in \|\Psi\|)\}$

iii. $\|[x_1,...,x_n][\varphi_1,...,\varphi_m]\|$
$= \{\langle a, a' \rangle \mid a\,[x_1 ... x_n]\,a' \;\&\; a' \in \|\varphi_1\| \cap ... \cap \|\varphi_m\|\}$

A DRS Φ is defined to be *true* in a model M under an assignment a iff there is some assignment a' such that $\langle a, a' \rangle \in \|\Phi\|^M$; a condition φ is *true* in M under a iff $a \in \|\varphi\|^M$.

Discourse Representation Theory comes with a set of construction rules that translate certain English texts into DRSs and thus provide these texts with an interpretation (see e.g. Kamp [1981] or Kamp & Reyle [to appear]). For example the DRS associated with text (2) is (3) and the DRS connected to (4) is (5).

(2) A farmer owns a donkey. He beats it.

(3) $[x_1, x_2][farmer\ x_1, donkey\ x_2, own\ x_2x_1, beat\ x_2x_1]$

(4) Every farmer who owns a donkey beats it.

(5) $[\,][[x_1, x_2][farmer\ x_1, donkey\ x_2, own\ x_2x_1] \Rightarrow [\,][beat\ x_2x_1]]$

---

[11] In this and the following definition I choose the transparent format of Groenendijk & Stokhof [1991].

The following function † gives an embedding of DRT into predicate logic, essentially the one that is discussed in Kamp & Reyle [to appear].

DEFINITION 4 (Translating DRT into predicate logic).

i.	$(Rt_1 \dots t_n)^\dagger$	=	$Rt_1 \dots t_n$
	$(t_1 = t_2)^\dagger$	=	$t_1 = t_2$
ii.	$(\neg\Phi)^\dagger$	=	$\neg\Phi^\dagger$
	$(\Phi \vee \Psi)^\dagger$	=	$\Phi^\dagger \vee \Psi^\dagger$
	$([x_1,\dots,x_n][\varphi_1,\dots,\varphi_m] \Rightarrow \Psi)^\dagger$	=	$\forall x_1 \dots x_n((\varphi_1{}^\dagger \wedge\dots\wedge \varphi_m{}^\dagger) \rightarrow \Psi^\dagger)$
iii.	$([x_1,\dots,x_n][\varphi_1,\dots,\varphi_m])^\dagger$	=	$\exists x_1 \dots x_n(\varphi_1{}^\dagger \wedge\dots\wedge \varphi_m{}^\dagger)$

A simple induction on the complexity of DRT constructs shows that a condition or DRS δ is true in M under a if and only if its translation $\delta^\dagger$ is. Note that the translation is sensitive to context: the translation of a condition $\Phi \Rightarrow \Psi$ is *not* given as a function of the translations of Φ and Ψ.

By way of example the images under † of the DRSs (3) and (5) are given in (6) and (7).

(6) $\exists x_1 x_2\, (\textit{farmer } x_1 \wedge \textit{donkey } x_2 \wedge \textit{own } x_2 x_1 \wedge \textit{beat } x_2 x_1)$

(7) $\forall x_1 x_2\, ((\textit{farmer } x_1 \wedge \textit{donkey } x_2 \wedge \textit{own } x_2 x_1) \rightarrow \textit{beat } x_2 x_1)$

We can go the other way around as well, translating predicate logic into the DRT language. The function * defined here will do the job.

DEFINITION 5 (Translating predicate logic into DRT).

i.	$(Rt_1 \dots t_n)^*$	=	$[\,][Rt_1 \dots t_n]$
	$(t_1 = t_2)^*$	=	$[\,][t_1 = t_2]$
ii.	$(\neg\varphi)^*$	=	$[\,][\neg\varphi^*]$
	$(\varphi \vee \psi)^*$	=	$[\,][\varphi^* \vee \psi^*]$
iii.	$(\exists x\varphi)^*$	=	$[x][\varphi^* \vee \varphi^*]$

Again it is easily seen that a formula φ is true in M under a if and only if the DRS φ^* is, and indeed that $\varphi^{*\dagger}$ is logically equivalent to φ. So in a sense, as far as truth conditions are concerned, DRT is a notational variant of ordinary predicate logic. It should be observed, however, that † in fact ignores one of the most important aspects of DRT, namely its dynamic character. It is here that DRT and predicate logic differ in expressive power. While a DRS characterizes a (binary) *relation* of assignments, formulas of predicate logic take *sets* of assignments as their values.

In order to compare DRT with the logical system described in the previous section I'll now give a translation of DRT into that system. Since the Kamp & Reyle function † that we have just discussed is sensitive to context and loses the dynamic aspect of DRT, it is worth considering a slightly more complicated embedding. This new translation will also generalise easier to translations of other systems of dynamic logic into type logic.

Note that the context states that were introduced in the previous section in an obvious way correspond to assignments. Even though for technical reasons we have decided to let states be primitive objects and to let stores be functions from states to appropriate values, we can intuïtively view states as functions from stores to values. This inspires us to define the following translation ° from DRT into our logic. We let the translation $(x_n)^\circ$ of the n-th variable of the DRT language (in some fixed ordering) be $v_n i$, where v_n is the n-th store name of type se (remember that store names are constants) and i is the first variable of type s. The translation $(c)^\circ$ of any individual constant c is just c itself. The translations of conditions and DRSs we get simply by copying the clauses in definition 3, sending conditions to closed type st terms and DRSs to closed terms of type $s(st)$ in the following way.

DEFINITION 6 (Translating DRT into the Theory of Types).

i.	$(Rt_1 \dots t_n)^\circ$	=	$\lambda i\,(Rt_1{}^\circ \dots t_n{}^\circ)$
	$(t_1 = t_2)^\circ$	=	$\lambda i\,(t_1{}^\circ = t_2{}^\circ)$
ii.	$(\neg\Phi)^\circ$	=	$\lambda i\,\neg\exists j\,(\Phi^\circ ij)$
	$(\Phi \vee \Psi)^\circ$	=	$\lambda i\exists j\,(\Phi^\circ ij \vee \Psi^\circ ij)$
	$(\Phi \Rightarrow \Psi)^\circ$	=	$\lambda i\forall j\,(\Phi^\circ ij \rightarrow \exists k\,\Psi^\circ jk)$
iii.	$([x_1,\dots,x_n][\varphi_1,\dots,\varphi_m])^\circ$	=	$\lambda ij\,(i\,[v_1,\dots,v_n]j \wedge \varphi_1{}^\circ j \wedge\dots\wedge \varphi_m{}^\circ j)$

Given our axioms AX1-AX3 definition 6 obviously does the same as definition 3 did, but now via a translation into type logic. Let us apply ° to the two example DRSs given above to see how things work out. It is not difficult to see that (3) gets a translation that—after some lambda conversions—turns out to be equivalent to the following term.

(8) $\lambda ij\,(i\,[v_1,v_2]j \wedge farmer\,(v_1 j) \wedge donkey\,(v_2 j) \wedge own\,(v_2 j)(v_1 j) \wedge beat(v_2 j)(v_1 j))$

This term denotes the relation that holds between two states if they differ maximally in two stores and in the second state the value of the first store is a farmer, while the value of the second store is a donkey that he owns and beats. Copying the definition of truth for DRSs, let's say that an $s(st)$ term Φ is *true* in a given state i if and only if there is some state j such that $\langle i, j\rangle$ is in the denotation of Φ. The set of all states i such that Φ is true in i— the *content* of Φ— we define as the denotation of $\lambda i\exists j\,(\Phi ij)$ (the domain of the relation Φ). This means that the content of (8) is the denotation of

(9) $\lambda i\exists j\,(i\,[v_1,v_2]j \wedge farmer\,(v_1 j) \wedge donkey\,(v_2 j) \wedge own\,(v_2 j)(v_1 j) \wedge beat(v_2 j)(v_1 j))$.

An application of the Unselective Binding lemma readily reduces this to the simpler

(10) $\lambda i \exists x_1 x_2 (farmer\ x_1 \wedge donkey\ x_2 \wedge own\ x_2 x_1 \wedge beat\ x_2 x_1)$,

which is just (6) preceded by a vacuous λi and which in an obvious sense gives the right truth conditions: the text is true in all states if (6) is true, false in all states if this sentence is false.

We get the translation of (5) in the following way. First we note that the subDRS $[x_1,x_2][farmer\ x_1,\ donkey\ x_2,\ own\ x_2 x_1]$ translates as

(11) $\lambda ij\ (i\ [v_1,v_2]j \wedge farmer\ (v_1 j) \wedge donkey\ (v_2 j) \wedge own\ (v_2 j)(v_1 j))$,

while $([][beat\ x_2 x_1])°$ equals $\lambda ij\ (i = j \wedge beat\ (v_2 j)(v_1 j))$. Now using clause ii. of definition 6, doing some lambda conversions and using predicate logic we find that (5)° is equivalent to

(12) $\lambda ij\ (i = j \wedge \forall k\ ((i\ [v_1, v_2]k \wedge farmer\ (v_1 k) \wedge donkey\ (v_2 k)$
$\wedge own\ (v_2 k)(v_1 k)) \rightarrow beat\ (v_2 k)(v_1 k))$,

which by Unselective Binding reduces to

(13) $\lambda ij\ (i = j \wedge \forall x_1 x_2 (farmer\ x_1 \wedge donkey\ x_2 \wedge own\ x_2 x_1) \rightarrow beat\ x_2 x_1))$,

and which has the following term—(7) preceded by a vacuous lambda abstraction—for its content.

(14) $\lambda i \forall x_1 x_2 (farmer\ x_1 \wedge donkey\ x_2 \wedge own\ x_2 x_1) \rightarrow beat\ x_2 x_1)$

We thus see that it is possible to provide the DRT fragment with a semantics by replacing definition 3 by our translation of DRSs into type theory. Kamp's construction rules will then send English discourses to DRSs, the function ° sends DRSs to type logical terms and the usual interpretation function for type logic sends these terms to objects in (higher order) models. In the next section we'll see how we can shortcut this process by by-passing the level of DRSs and sending English expressions to logical terms directly.

But before we do this let me discuss the semantics of one more logical system, since we'll be able to borrow some ideas from this logic as well. The logic that I have in mind is the Dynamic Logic of Pratt [1976] (for a survey see Harel [1984], for a good introduction Goldblatt [1987], for connections with natural language Van Benthem [1989] and Groenendijk & Stokhof [1991]).

Dynamic Logic was set up in order to study aspects of the behaviour of (nondeterministic) computer programs. Like the expressions of DRT the expressions of dynamic logic are of two kinds: where DRT has conditions dynamic logic has *formulae* and where DRT has DRSs dynamic logic has *programs.* The following definition characterizes the syntax of QDL$^{\pm}$, a version of Pratt's Quantificational Dynamic Logic.[12]

DEFINITION 7 (QDL$^{\pm}$ Syntax).

i. If R is an n- ary relation constant and $t_1,...,t_n$ are terms, then $Rt_1...t_n$ is a formula.
   If t_1 and t_2 are terms then $t_1 = t_2$ is a formula.
   $\bot$ is a formula.

ii. If Φ is a program and φ is a formula then $[\Phi]\varphi$ is a formula.

iii. If x is a variable and t is a term then $x := ?$ and $x := t$ are programs.

iv. If φ is a formula then $\varphi?$ is a program.

v. If Φ and Ψ are programs then $\Phi\ ;\ \Psi$ is a program.

Intuïtively these constructions are meant to have the following meanings:

$[\Phi]\varphi$	after every terminating execution of Φ, φ is true
$x := ?$	non-deterministically assign some arbitrary value to x
$x := t$	assign the current value of t to x
$\varphi?$	test φ: continue if φ is true, otherwise fail
$\Phi\ ;\ \Psi$	do Φ and then Ψ

The meaning of a program is viewed as a relation between (program) states, each program state being some assignment of values to the program's variables. The idea is that a pair of program states $\langle a, a'\rangle$ is an element of the denotation of a given program iff starting in state a after execution the program could be in state a'. Since we study non-deterministic programs here, the binary relations under consideration need not be functions.

More formally, we can interpret the constructs of QDL$^{\pm}$ on first-order models M, sending programs Φ to binary relations $\|\Phi\|$ of assignments and formulae φ to sets of assignments $\|\varphi\|$ in the following way.

DEFINITION 8 (QDL$^{\pm}$ Semantics).

---

12 I write QDL$^{\pm}$ because the definition on the one hand gives a slight extension of Quantificational Dynamic Logic but on the other omits two clauses. Usually clause iv is restricted to read: if φ is an *atomic* formula then $\varphi?$ is a program. Since $\varphi?$ is interpreted as a program that tests whether φ is true, the restriction is reasonable on a computational interpretation. Note that our translation of DRT into QDL$^{\pm}$ given below depends on the extension to arbitrary formulae here. The clauses that are omitted here are those for *choice* and *iteration.* My QDL$^{\pm}$ is Groenendijk & Stokhof's [1991] QDL, except for the treatment of the assignment $x := t$. Contrary to what Groenendijk & Stokhof assume this atomic program cannot be defined as $x := ?\ ;\ x = t\ ?$ (consider $x := x$, for example).

$$
\begin{array}{llcl}
\text{i.} & \|Rt_1 \ldots t_n\| & = & \{a \mid \langle \|t_1\|^a, \ldots, \|t_n\|^a \rangle \in I(R)\} \\
 & \|t_1 = t_2\| & = & \{a \mid \langle \|t_1\|^a = \|t_2\|^a \rangle\} \\
 & \|\bot\| & = & \varnothing \\
\text{ii.} & \|[\Phi]\varphi\| & = & \{a \mid \forall a' (\langle a, a' \rangle \in \|\Phi\| \rightarrow a' \in \|\varphi\|)\} \\
\text{iii.} & \|x := ?\| & = & \{\langle a, a' \rangle \mid a[x]a'\} \\
 & \|x := t\| & = & \{\langle a, a' \rangle \mid a[x]a' \ \&\ a'(x) = \|t\|^a\} \\
\text{iv.} & \|\varphi ?\| & = & \{\langle a, a \rangle \mid a \in \|\varphi\|\} \\
\text{v.} & \|\Phi ; \Psi\| & = & \{\langle a, a' \rangle \mid \exists a'' (\langle a, a'' \rangle \in \|\Phi\| \ \&\ \langle a'', a' \rangle \in \|\Psi\|)\}
\end{array}
$$

In the last clause $\|\Phi ; \Psi\|$ is defined to be the composition of the relations $\|\Phi\|$ and $\|\Psi\|$. Note that $[\Phi]\varphi$ is in fact a modal statement; the interpretation of Φ giving the relevant accessibility relation.

Quantificational Dynamic Logic in this formulation subsumes predicate logic. In particular, we can consider $\varphi \rightarrow \psi$ and $\forall x\varphi$ to be abbreviations of $[\varphi ?]\psi$ and $[x := ?]\varphi$ respectively. From $\rightarrow$ and $\bot$ we can of course define all other propositional connectives in the usual way.

That the logic subsumes DRT as well we can show by interpreting conditions as if they were abbreviations for certain QDL$^\pm$ formulae and DRSs as shorthand for certain programs. The following function ‡ preserves meaning.

DEFINITION 9 (Translating DRT into QDL$^\pm$).

$$
\begin{array}{llcl}
\text{i.} & (Rt_1 \ldots t_n)^\ddagger & = & Rt_1 \ldots t_n \\
 & (t_1 = t_2)^\ddagger & = & t_1 = t_2 \\
\text{ii.} & (\neg\Phi)^\ddagger & = & [\Phi^\ddagger]\bot \\
 & (\Phi \vee \Psi)^\ddagger & = & \neg([\Phi^\ddagger]\bot \wedge [\Psi^\ddagger]\bot) \\
 & (\Phi \Rightarrow \Psi)^\ddagger & = & [\Phi^\ddagger]\neg[\Psi^\ddagger]\bot \\
\text{iii.} & ([x_1, \ldots, x_n][\varphi_1, \ldots, \varphi_m])^\ddagger & = & x_1 := ? ; \ldots ; x_1 := ? ; \varphi_1^\ddagger ?; \ldots ; \varphi_m^\ddagger ?
\end{array}
$$

A simple induction shows that $\|\delta\| = \|\delta^\ddagger\|$ for any condition or DRS δ.

A few examples may show that analysing natural language with the help of QDL$^\pm$ rather than DRT can be advantageous. Let's consider (2) again (here given as (15)). Its DRT rendering was (16); the translation under ‡ of this DRS is (17). This program is pretty close to the original text: each atomic program in (17) corresponds to a word in (15) (the two random assignments match with the indefinite articles), each word in (15), except the two pronouns, corresponds to an atomic program in (17). Moreover, each of the two sentences in the little text matches with a constituent of (17): the first sentence with the first five atomic programs, the second sentence with the last one. We can get the translation of the text simply by sequencing the translations of its constituent sentences. Here the QDL$^\pm$ program fares better as its DRT equivalent since (16) cannot be split into two separate constituents.

(15) A farmer owns a donkey. He beats it.

(16) $[x_1, x_2][farmer\ x_1,\ donkey\ x_2,\ own\ x_2x_1,\ beat\ x_2x_1]$
(17) $x_1 := ?\ ;\ x_2 := ?\ ;\ farmer\ x_1\ ?\ ;\ donkey\ x_2\ ?\ ;\ own\ x_2x_1\ ?\ ;\ beat\ x_2x_1\ ?$
(18) $x_1 := ?\ ;\ farmer\ x_1\ ?\ ;\ x_2 := ?\ ;\ donkey\ x_2\ ?\ ;\ own\ x_2x_1\ ?\ ;\ beat\ x_2x_1\ ?$

When we consider the equivalent (18) we even get a bit closer to the text since there are now constituents to match the two indefinite NPs in the first sentence as well. The subprogram $x_1 := ?\ ;\ farmer\ x_1\ ?$ corresponds to *a farmer* and $x_2 := ?\ ;\ donkey\ x_2\ ?$ to *a donkey.*

Thus we see that dynamic logic can sometimes keep quite near to the form of an English text, but there are also indications that we might need its greater expressibility in order to be able to formalise texts correctly. Partee [1984], following a suggestion of Ewan Klein, uses assignments to model the behaviour of the moving reference point in linear narrative. Partee's paper is written in the DRT format, but strictly speaking assignments are not available within that framework. The idea can easily be expressed in dynamic logic however. Consider text (19), a linear narrative in which the actions are interpreted consecutively. It can be formalized as (20).

(19) A man entered a bar. He found a chair. He sat down.

(20) $x_1 := ?\ ;\ man\ x_1\ ?\ ;\ x_2 := ?\ ;\ bar\ x_2\ ?;\ enter\ x_2x_1r\ ?\ ;\ h := r\ ;\ r := ?\ ;$
$h \lesssim r\ ?\ ;$
$x_3 := ?\ ;\ chair\ x_3\ ?\ ;\ find\ x_3x_1r\ ?\ ;\ h := r\ ;\ r := ?\ ;\ h \lesssim r\ ?\ ;$
$sitdown\ x_1r\ ?\ ;\ h := r\ ;\ r := ?\ ;\ h \lesssim r\ ?$

Here each verb is evaluated with respect to a current reference time r (for example $find\ x_3x_1r$ means that x_1 finds x_3 at reference time r). Moreover, the evaluation of each verb causes the reference time to move forward. This is achieved by the subprogram $h := r\ ;\ r := ?\ ;\ h \lesssim r\ ?$ which first assigns the value of r to a help variable h, then makes a random assignment to r, and then tests whether the value of r is 'just after' the value of h, i.e. r's original value ($h \lesssim r$ means that r is just after h); the net effect is that r is nondeterministically shifted to a place just after its original position.[13] Note that again the program can be split into parts that each correspond to a sentence in the original text. If a sentence is added, the translation of the new text simply becomes the translation of the old text sequenced with the translation of that new sentence.

The following definition, a translation of QDL$^{\pm}$ into type logic, has much in common with our translation of DRT into this logic. For each term t we let t° be as before.

[13] For the moment we ignore that the reference point r should be situated before the point of speech. This will be taken into account in the theory that is sketched in section 7.

DEFINITION 10 (Translating QDL$^{\pm}$ into the Theory of Types).[14]

i.	$(Rt_1 \ldots t_n)^{\%}$	$=$	$\lambda i\,(Rt_1{}^{\circ} \ldots t_n{}^{\circ})$
	$(t_1 = t_2)^{\%}$	$=$	$\lambda i\,(t_1{}^{\circ} = t_2{}^{\circ})$
	$(\bot)^{\%}$	$=$	$\lambda i\ \bot$
ii.	$([\Phi]\varphi)^{\%}$	$=$	$\lambda i\ \forall j\,(\Phi^{\%}ij \rightarrow \varphi^{\%}j)$
iii.	$(x_n := ?)^{\%}$	$=$	$\lambda ij\,(i\,[v_n]j)$
	$(x_n := t)^{\%}$	$=$	$\lambda ij\,(i\,[v_n]j \wedge vj = t^{\circ})$
iv.	$(\varphi\,?)^{\%}$	$=$	$\lambda ij\,(i = j \wedge \varphi^{\%}i)$
v.	$(\Phi\,;\Psi)^{\%}$	$=$	$\lambda ij\exists k\,(\Phi^{\%}ik \wedge \Psi^{\%}kj)$

Again the embedding truthfully mirrors the definition of the semantics of the source language given the axoims AX1-AX3. We say that a program Ψ *follows from* a program Φ if and only if $\|\Phi\| \subseteq \|\Psi\|$ in all models. It is not difficult to prove that Ψ follows from Φ in QDL$^{\pm}$ if and only if AX1-AX3 $\models \forall ij\,(\Phi^{\%}ij \rightarrow \Psi^{\%}ij)$ in our type logic.

Let us see what effect % has on (20). It is easily seen that its three constituent parts, (20^a), (20^b) and (20^c), are translated to terms that are equivalent to (21), (22) and (23) respectively.

(20^a) $x_1 := ?\,;\, man\ x_1\,?\,;\, x_2 := ?\,;\, bar\ x_2\,?;\, enter\ x_2x_1r\,?\,;\, h := r\,;\, r := ?\,;$
$h \lesssim r\ ?$

(21) $\lambda ij\,(i\,[v_1,v_2,H,R]j \wedge man\,(v_1j) \wedge bar\,(v_2j) \wedge enter\,(v_2j)(v_1j)(Ri) \wedge Hj =$
$Ri \wedge Ri \lesssim Rj)$

(20^b) $x_3 := ?\,;\, chair\ x_3\,?\,;\, find\ x_3x_1r\,?\,;\, h := r\,;\, r := ?\,;\, h \lesssim r\ ?$

(22) $\lambda ij\,(i\,[v_3,H,R]j \wedge chair\,(v_3j) \wedge find\,(v_3j)(v_1j)(Ri) \wedge Hj = Ri \wedge Ri \lesssim Rj)$

(20^c) $sitdown\ x_1r\,?\,;\, h := r\,;\, r := ?\,;\, h \lesssim r\ ?$

(23) $\lambda ij\,(i\,[H, R]j \wedge sitdown\,(v_1j)(Ri) \wedge Hj = Ri \wedge Ri \lesssim Rj)$

In order to obtain the complete translation of (20) we must apply clause v of definition 10 twice. Sequencing (21) and (22) we get a term that after some tedious reductions[15] turns out to be equivalent to (24).

---

[14] For the translations of *choice* and *iteration* see Muskens [1991].

[15] Sequencing gives

$\lambda ij\exists k\,(i\,[v_1,v_2,H,R]k \wedge man\,(v_1k) \wedge bar\,(v_2k) \wedge enter\,(v_2k)(v_1k)(Ri) \wedge Hk = Ri \wedge Ri \lesssim Rk \wedge$
$k\,[v_3,H,R]j \wedge chair\,(v_3j) \wedge find\,(v_3j)(v_1j)(Rk) \wedge Hj = Rk \wedge Rk \lesssim Rj)$.

Use the definition of $k\,[v_3,H,R]j$ to write this as

(24) $\lambda ij\exists t_1 (i\,[v_1,v_2,v_3,H,R\,]j \wedge man\,(v_1 j\,) \wedge bar\,(v_2 j\,) \wedge enter\,(v_2 j\,)(v_1 j\,)(Ri\,) \wedge chair\,(v_3 j\,) \wedge find\,(v_3 j\,)(v_1 j\,)t_1 \wedge Hj = t_1 \wedge Ri \lesssim t_1 \lesssim Rj\,)$

Reductions of a similar kind show that the result of sequencing (24) with (23) is equivalent to (25).

(25) $\lambda ij\exists t_1 t_2 (i\,[v_1,v_2,v_3,H,R\,]j \wedge man\,(v_1 j\,) \wedge bar\,(v_2 j\,) \wedge enter\,(v_2 j\,)(v_1 j\,)(Ri\,) \wedge chair\,(v_3 j\,) \wedge find\,(v_3 j\,)(v_1 j\,)t_1 \wedge sitdown\,(v_1 j\,)t_2 \wedge Hj = t_2 \wedge Ri \lesssim t_1 \lesssim t_2 \lesssim Rj\,)$

We see the reference time *Rj* move forward here. Each part of the text has an input reference time *Ri* and an output reference time *Rj.* If a sentence is linked to the right of a text, its input reference time *Ri* will pick up the output reference time of the text, while its output reference time *Rj* provides the reference time for possible continuations. The following term gives the content of (25).[16]

(26) $\lambda i\exists x_1 x_2 x_3 \exists t_1 t_2 t_3 (man\ x_1 \wedge bar\ x_2 \wedge enter\ x_2 x_1 (Ri\,) \wedge chair\ x_3 \wedge find\ x_3 x_1 t_1 \wedge sitdown\ x_1 t_2 \wedge Ri \lesssim t_1 \lesssim t_2 \lesssim t_3\,)$

Notice that not all store names have disappeared. The text can only be evaluated with respect to a given input reference time, therefore *R* should appear as a parameter whose value is to be provided by the input context state.

5. MORE DONKEY BUSINESS

The embedding ° given in the previous section shows that it possible to combine the dynamics of Discourse Representation Theory with the logical engine behind Montague Grammar. I now want to cash in on this insight. I'll define a little fragment of English that has possibilities for anaphoric linkage and translate it into type logic, thus providing the fragment with a semantics. The treatment will resemble the theories

---

$\lambda ij\exists k\,(i\,[v_1,v_2,H,R\,]k \wedge k\,[v_3,H,R\,]j \wedge man\,(v_1 j\,) \wedge bar\,(v_2 j\,) \wedge enter\,(v_2 j\,)(v_1 j\,)(Ri\,) \wedge Hk = Ri \wedge chair\,(v_3 j\,) \wedge find\,(v_3 j\,)(v_1 j\,)(Rk\,) \wedge Hj = Rk \wedge Ri \lesssim Rk \lesssim Rj\,)$.

Given our axioms this last term is equivalent to

$\lambda ij\exists h\,(i\,[v_1,v_2,v_3,H,R\,]j \wedge i\,[R\,]h \wedge man\,(v_1 j\,) \wedge bar\,(v_2 j\,) \wedge enter\,(v_2 j\,)(v_1 j\,)(Ri\,) \wedge chair\,(v_3 j) \wedge find\,(v_3 j\,)(v_1 j\,)(Rh\,) \wedge Hj = Rh \wedge Ri \lesssim Rh \lesssim Rj\,)$,

which can be reduced to (24) with the help of Unselective Binding.

16 Clearly a suitable assumption on the ordering $\lesssim$ would allow us to get rid of t_3 in this term.

given by Kamp and Heim in the sense that it makes the same predictions as these theories do, but I'll work in Montague's way and shall employ nothing beyond the resources of ordinary type logic and the axioms given in section 3. For the moment I shall only consider nominal anaphora, but the treatment of temporal anaphora in section 7 below will extend the fragment that is given here. I'll use a categorial grammar that is based on a set of categories generated by the two following rules.

i. S and E are categories;
ii. If A and B are categories, then $A /^n B$ and $B \backslash^n A$ are categories ($n \geq 1$).

Here S is the category of sentences (and texts). The category E does not itself correspond to any class of English expressions, but is used to build up complex categories that do correspond to such classes. The notations $/^n$ and $\backslash^n$ stand for sequences of n slashes (the possibility to have multiple slashes will not be used until section 7). I write

N	(common noun phrase)	for	S / E,
NP	(noun phrase)	for	$S / (E \backslash S)$,
VP	(verb phrase)	for	$E \backslash S$,
TV	(transitive verb phrase)	for	VP / NP, and
DET	(determiner)	for	NP / N.

Table 1 below gives the lexicon of our toy grammar. Each basic expression of the fragment is assigned to a category. From basic expressions complex expressions can be built. An expression of category $A /^n B$ ($B \backslash^n A$) followed (preceded) by an expression of category B forms an expression of category A. For example, the word a_3 of category DET (defined as NP / N) combines with the word *man* of category N to the phrase a_3 *man,* which belongs to the category NP. The word *see* of category TV can then combine with a_3 *man* to the verb phrase *see* a_3 *man.* I counterfactually assume that agreement phenomena have been taken care of, so that the combination of the NP $John_0$ with the verb phrase *see* a_3 *man* is written as the sentence $John_0$ *sees* a_3 *man.*

Category	*Type*	*Some basic expressions*
VP	$[e]$	*walk, talk*
N	$[e]$	*farmer, donkey, man, woman, bastard*
NP	$[[e]]$	*I, you,* $Mary_n$, $John_n$, it_n, he_n, she_n $(n \geq 0)$
TV	$[[[e]]e]$	*own, beat, love, see*
DET	$[[e][e]]$	a_n, $every_n$, the_n $(n \geq 0)$
$(N \backslash N)/VP$	$[[e][e]e]$	*who*
$S \backslash (S/S)$	$[[][]]$	*and, or, .* (the stop)
$(S/S)/S$	$[[][]]$	*if*

Table 1.

Determiners, proper names and pronouns in the fragment are randomly indexed. Coindexing is meant to indicate the relation between a dependent (for example an anaphoric pronoun) and its antecedent. I assume that some form of the Binding Theory is used to rule out undesired coindexings such as in *$John_0$ sees him_0*, but I shan't take the trouble to spell out the relevant rules.

Since sentences and texts are treated as relations between states, we'll associate type $s(st)$ with category S. Type e is associated with category E. The type associated with a complex category $A/^n B$ or $B \backslash^n A$, is $(TYP(B), TYP(A))$, where $TYP(B)$ is the type associated with B and $TYP(A)$ is the type associated with A. This means that an expression that seeks an expression of category B in order to combine with it into an expression of category A is linked with a function from $TYP(B)$ objects to $TYP(A)$ objects. Thus our category-to-type rule is

i. $TYP(S) = s(st)$; $TYP(E) = e$;
ii. $TYP(A/^n B) = TYP(B \backslash^n A) = (TYP(B), TYP(A))$.

To improve readability let's write $[\alpha_1 \ldots \alpha_n]$ for $(\alpha_1(\alpha_2(\ldots\alpha_n(s(st)))\ldots)$. The category-to-type rule assigns the types that are listed in the second column of Table 1 to the categories listed in the first column. A type $[\alpha_1 \ldots \alpha_n]$ can be thought of as a *stack;* α_1 is at the top, application (to a type α_1 object) is popping the stack, abstraction is pushing it.

We now come to the translation into type theory of our little fragment of English. Expressions of a category A will be translated into terms of type $TYP(A)$ by an inductive definition that gives the translations of basic expressions and says how the translation of a complex expression depends on the translations of its parts. This last combination rule is easily stated: if σ is a translation of the expression Σ of category $A/^n B$ or $B \backslash^n A$ and if ξ translates the expression Ξ of category B, then the translation of the result of combining Σ and Ξ will be the term $\sigma\xi$. In other words, combination will always correspond to functional application.

In the translations of basic expressions I shall let *h, i, j, k* and *l* be type *s* variables; *x* and *y* type *e* variables; (subscripted) *P* a variable of type *TYP* (*VP*); *Q* a variable of type *TYP* (*NP*); *p* and *q* variables of type *s* (*st*); *mary* a constant of type *e* ; *farmer* and *walk* type *et* constants; *love* a constant of type *e* (*et*) and *speaker, addressee* and each v_n store names of type *se*.

Conjunction of sentences is formalised as sequencing, i.e. composition of relations. (Compare clause v. of definition 10.)

and	$\rightsquigarrow$	$\lambda pq\lambda ij\exists h\,(pih \wedge qhj)$
.	$\rightsquigarrow$	$\lambda pq\lambda ij\exists h\,(pih \wedge qhj)$

The translation of the indefinite determiner *a*$_n$ will be a term that searches for predicates P_1 and P_2 as usual. If particular choices for P_1 and P_2 are plugged in, a program results that consists of three parts: first a random assignment is made to v_n (compare clause iii. of definition 10); then the program that is the result of applying P_1 to the value of v_n is carried out and after that the result of applying P_1 to the value of v_n is executed.

a$_n$	$\rightsquigarrow$	$\lambda P_1P_2\lambda ij\exists kh\,(i\,[v_n]k \wedge P_1\,(v_nk)kh \wedge P_2\,(v_nk)hj)$

We let simple verbs and nouns essentially act as tests (compare clause iv. of definition 10).

farmer	$\rightsquigarrow$	$\lambda x\lambda ij\,(i = j \wedge farmer\, x)$
walk	$\rightsquigarrow$	$\lambda x\lambda ij\,(i = j \wedge walk\, x)$
own	$\rightsquigarrow$	$\lambda Q\lambda y\,(Q\lambda x\lambda ij\,(i = j \wedge own\, xy))$

These basic translations provide us with enough material to translate our first sentences. The reader may wish to verify e.g. that the sentence *a*$_1$ *farmer owns a*$_2$ *donkey* translates as:[17]

(27) $\lambda ij\,(i\,[v_1,v_2]j \wedge farmer\,(v_1j) \wedge donkey\,(v_2j) \wedge own\,(v_2j)(v_1j))$.

We see here that indefinites create discourse referents. Definites, on the other hand are only able to pick up referents. Therefore the translation of the determiner *the*$_n$, given below, differs from the translation of *a*$_n$ in that the random assignment to v_n has been skipped. For the rest the translation is similar: first the program that is the result of

[17] Of course *donkey* translates as $\lambda x\lambda ij\,(i = j \wedge donkey\ x)$. Here and in the rest of the paper I shall adopt the convention that a basic translation will not be explicitly given if it can easily be read off from some obvious paradigm case.

applying P_1 to the value of v_n is carried out and then the result of applying P_2 to the value of v_n is executed. The translations of the proper name *Mary*$_n$ and the pronoun *it*$_n$ involve only one predicate. The first translation can be understood as the translation of *the*$_n$ applied to the predicate 'be Mary', $\lambda x \lambda ij\,(i = j \wedge x = mary)$, and the translation of *it*$_n$ can be understood as the translation of *the*$_n$ applied to the skip predicate $\lambda x \lambda ij\,(i = j)$.

the$_n$	$\rightsquigarrow$	$\lambda P_1 P_2 \lambda ij \exists k\,(P_1(v_n k)ik \wedge P_2(v_n k)kj)$
Mary$_n$	$\rightsquigarrow$	$\lambda P \lambda ij\,(v_n i = mary \wedge P(v_n i)ij)$
it$_n$	$\rightsquigarrow$	$\lambda P \lambda ij\,(P(v_n i)ij)$

Using these translations we find e.g. that the sentence *the*$_1$ *bastard beats it*$_2$ translates as:

(28) $\lambda ij\,(i = j \wedge bastard\,(v_1 i) \wedge beat\,(v_2 i)(v_1 i))$.

We can now combine the two sentences into the text *a*$_1$ *farmer owns a*$_2$ *donkey. the*$_1$ *bastard beats it*$_2$, whose translation we obtain by sequencing (27) and (28). The result is (29), which has (30) for its content.

(29) $\lambda ij\,(i\,[v_1, v_2]\,j \wedge farmer\,(v_1 j) \wedge donkey\,(v_2 j) \wedge own\,(v_2 j)(v_1 j)$
$\wedge\, bastard\,(v_1 j) \wedge beat\,(v_2 j)(v_1 j))$.

(30) $\lambda i \exists xy\,(farmer\,x \wedge donkey\,y \wedge own\,yx \wedge bastard\,x \wedge beat\,yx)$.

We see that the definites *the*$_1$ and *it*$_2$ succeed in picking up the referents that were introduced in the first sentence. On the other hand, if *the*$_1$ *bastard beats it*$_2$ is interpreted without a previous introduction of the two relevant discourse referents the context must provide for them: the text will only be true in context states i such that $v_1 i$ is a bastard that beats $v_2 i$ then. This is the deictic use of definites.

Other modes of combination are possible as well. Let us translate the word *if* as follows. The word requires two arguments p and q. If *if* is applied to particular p and q, a program results that tests whether $p \Rightarrow q$ is true in the current state, continues if the answer is yes, but fails if the answer is no (compare clause ii. of definition 6 and clause iv. of definition 10). In a similar way the translation of *or* tests whether one of the disjuncts is true.

if	$\rightsquigarrow$	$\lambda pq \lambda ij\,(i = j \wedge \forall h\,(pih \rightarrow \exists k\,qhk))$
or	$\rightsquigarrow$	$\lambda pq \lambda ij\,(i = j \wedge \exists h\,(pih \vee qih))$

Plugging in (27) and (28) into the translation of *if* after reductions gives:

(31) $\lambda ij\,(i = j \wedge \forall xy\,((farmer\,x \wedge donkey\,y \wedge own\,yx) \rightarrow (bastard\,x \wedge beat\,yx)))$,

the translation of *if a_1 farmer owns a_2 donkey the_1 bastard beats it_2*.[18]

Here too the definites succeeded in picking up the relevant discourse referents, but note that these referents are no longer available once (31) is processed. The translation of this sentence acts as a *test* ; it cannot change the value of any store but can only serve to rule out certain continuations of the interpretation process. The discourse referents that were introduced by the determiners a_1 and a_2 had a limited life span. Their role was essential in obtaining a correct translation of the sentence, but once this translation was obtained they died and could no longer be accessed.

The translation of *$every_1$ farmer who owns a_2 donkey beats it_2* becomes available as soon as we have translations for the words *who* and *$every_n$*. These are defined as follows.

who $\leadsto$ $\lambda P_1 P_2 \lambda x \lambda ij \exists h\,(P_2 xih \wedge P_1 xhj)$
$every_n$ $\leadsto$ $\lambda P_1 P_2 \lambda ij\,(i = j \wedge \forall kl\,((i\,[v_n]k \wedge P_1\,(v_n k)kl) \rightarrow \exists h\,P_2\,(v_n k)lh))$

In fact the word *who* dynamically conjoins two predicates and the translation of *$every_n$* is a variation upon the translation of *if.* The reader is invited to check that the famous donkey sentence translates as (32).

(32) $\lambda ij\,(i = j \wedge \forall xy\,((farmer\,x \wedge donkey\,y \wedge own\,yx) \rightarrow beat\,yx))$.

We have seen that definites can either be used anaphorically, picking up referents that were introduced earlier in the discourse, or deictically, putting restrictions on the initial context state. Here are translations for two words that can only be used deictically.

I $\leadsto$ $\lambda P \lambda ij\,(P\,(speaker\,i)ij)$
you $\leadsto$ $\lambda P \lambda ij\,(P\,(addressee\,i)ij)$

For example, *the_1 girl loves you* is now rendered as (33), which has (34) for its content. The context must provide a particular girl and a particular addressee for the text to be true or false.

---

[18] Note that a sentence like *if $Mary_1$ owns a_2 donkey she_1 beats it_2* is predicted to be *true* in those states in which the value of v_1 is not Mary. This is not very satisfying; however, if we would let the sentence *$Mary_1$ owns a_2 donkey* presuppose, rather than assert, the statement $v_1 i = mary$, the latter statement would be a presupposition of the whole conditional as well. In the context of this paper there is hardly any reason to develop a full theory of presuppositions, but see Van Eijck [1991] for an approach that may well be compatible with the present theory.

(33) $\lambda ij\,(i = j \;\wedge\; girl\,(v_1 i) \;\wedge\; love\,(addressee\; i)(v_1 i))$
(34) $\lambda i\,(girl\,(v_1 i) \;\wedge\; love\,(addressee\; i)(v_1 i))$

6. TEMPORAL ONTOLOGY

Our models have enough structure now to get the dynamics going, but we need some more structure to be able to interpret the English tenses. In this section I'll impose the necessary temporal ontology. A special tense logic, or a logic with a special tense component (such as Montague's **IL**) is not needed, however, since with the help of some axioms we can simply ensure that the ground domains of our models provide us with as much structure as is required. We don't need a tense logic to treat the tenses, just as we don't need a dynamic logic in order to handle the dynamics of language.

There are many ways to define the necessary structure, all of them compatible with the dynamics that we have introduced. Here I shall assume a rather rich ontology, consisting of *possible eventualities, periods of time* and *possible worlds.* For each of these basic ingredients we'll have a special ground type; type ε for eventualities, type τ for periods of time and type w for worlds. Accordingly, the basic domain D_ε will be a set of eventualities, D_τ will be a set of periods and D_w a set of worlds.

Let's consider periods of time. It is natural to order these by a relation of *complete temporal precedence.* We use $<$, a constant of type $\tau(\tau t)$, to express this precedence relation and define four other useful relations in terms of it. We let

$t_1 \subseteq t_2$	abbreviate	$\forall t\,(t_2 < t \rightarrow t_1 < t) \wedge \forall t\,(t < t_2 \rightarrow t < t_1)$,	
$t_1 \,\mathsf{O}\, t_2$	abbreviate	$\exists t\,(t \subseteq t_1 \wedge t \subseteq t_2)$,	
$t_1 << t_2$	abbreviate	$\forall t\,(t_2 < t \rightarrow t_1 < t)$	and
$t_1 \lessapprox t_2$	abbreviate	$t_1 < t_2 \wedge \neg\exists t_3\; t_1 < t_3 < t_2$.	

The first two of these definitions are borrowed from Van Benthem [1983]. Note that the definitions have as a consequence that $\subseteq$, *temporal inclusion,* is reflexive and transitive, that O, *temporal overlap,* is reflexive and symmetric and that $<<$ is reflexive and transitive.

I assume the following five temporal axioms.

AX4	$\forall t\; \neg t < t$
AX5	$\forall t_1 t_2 t_3\,((t_1 < t_2 \wedge t_2 < t_3) \rightarrow t_1 < t_3)$
AX6	$\forall t_1 t_2 t_3\,(t_1 < t_2 \vee t_1 \,\mathsf{O}\, t_2 \vee t_2 < t_1)$
AX7	$\forall t_1 t_2 t_3\,((t_1 \subseteq t_2 \wedge t_2 \subseteq t_1) \rightarrow t_1 = t_2)$
AX8	$\forall t_1 \exists t_2\; t_1 \lessapprox t_2 \wedge \forall t_1 \exists t_2\; t_2 \lessapprox t_1$

The first two axioms simply state that temporal precedence is a strict partial order, the third says that any two periods are comparable: either they overlap, or one of the two precedes the other. The fourth axiom gives us antisymmetry for the inclusion relation $\subseteq$, and thus makes $\subseteq$ into a partial ordering. The last axiom, which is useful for technical reasons, states that any period is immediately followed by another and immediately preceded by one. Some elementary reasoning shows that AX5 and AX6, in conjunction with the definitions that were given above, entail $\forall t_1 t_2 (t_1 << t_2 \vee t_2 << t_1)$.

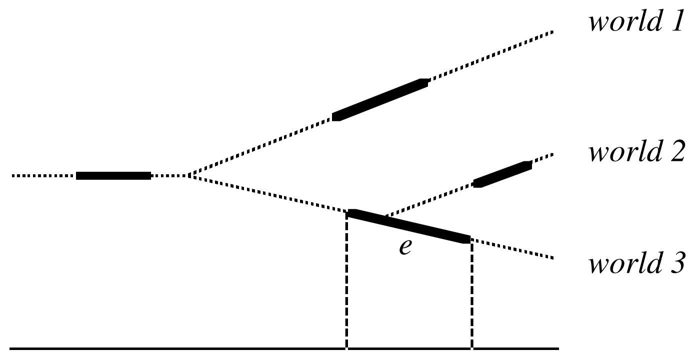
The intuition behind AX4-AX8 is that we view periods of time as segments of a Euclidean straight line and that we interpret $<$ as 'lying completely to the left of'. Under this interpretation, $\subseteq$ is inclusion of segments, O is having a segment in common, $t_1 << t_2$ means that t_2's end point is not to the left of t_1's, and $t_1 \lessapprox t_2$ means that t_1's end point coincides with t_2's start. The axioms given here do not entail everything that is true under this geometrical interpretation (for example, we cannot derive that for any two overlapping periods there is a period that is their intersection), but I consider anything that is true under the interpretation acceptable as an axiom for our time structures.

Eventualities differ from periods of time in several ways. Firstly, two eventualities that occur simultaneously need not be identical, while a period is completely determined by its temporal relations to other periods (AX7 ensures this). Secondly, eventualities are contingent, but periods are not. For example, while it is completely sure now, in May, that there will be a next month of June (a period), it is still a contingent matter whether I shall make a trip to Portugal (an eventuality) in that month. The future is contingent. On the other hand, everything that has happened last March is fixed now and can no longer be altered. Therefore, while periods are ordered as segments on a line, we may view the relation of precedence on the set of eventualities as branching in the direction of the future. Each eventuality has a unique past, but it may have many possible futures. Dowty [1979] has pointed out that we need this kind of branching if we want to avoid the so-called Imperfective Paradox, a puzzle that arises in connection with the semantics of the progressive. We shall deal with the progressive below and shall avoid the Imperfective Paradox in Dowty's way.

The picture that I have in mind looks as follows.[19] Periods of time are ordered like the segments of a straight line. Eventualities have a branching ordering. We can associate with each eventuality e the period ϑe at which it takes place (hence ϑ must be a function of type $\varepsilon\tau$). Each eventuality occurs in many possible worlds since each has many possible futures. In fact, worlds could be construed as maximal chains (as branches) in the branching precedence ordering of eventualities.

---

19 Leonoor Oversteegen's *Two Track Theory of Time* (see Oversteegen [1989]) partly inspired me to consider the structures in this section. My D_ε resembles Oversteegen's 'E track' and my D_τ her 'S track'.



However, we shall let worlds be primitive and define the precedence relation on eventualities with their help. Let *in* be a constant of type $\varepsilon(w\ t)$. We say that eventualities e_1 and e_2 are *comparable* if and only if $\exists w\,(e_1\ in\ w \wedge e_2\ in\ w)$. Each branch in the structure of eventualities now inherits the relations that were defined on the domain of periods. We write

$e_1 < e_2$	for	$\vartheta e_1 < \vartheta e_2 \wedge \exists w\,(e_1\ in\ w \wedge e_2\ in\ w)$,	
$e_1 \subseteq e_2$	for	$\vartheta e_1 \subseteq \vartheta e_2 \wedge \exists w\,(e_1\ in\ w \wedge e_2\ in\ w)$,	
$e_1\ \mathrm{O}\ e_2$	for	$\vartheta e_1\ \mathrm{O}\ \vartheta e_2 \wedge \exists w\,(e_1\ in\ w \wedge e_2\ in\ w)$,	
$e_1 << e_2$	for	$\vartheta e_1 << \vartheta e_2 \wedge \exists w\,(e_1\ in\ w \wedge e_2\ in\ w)$,	and
$e_1 \lesssim e_2$	for	$\vartheta e_1 \lesssim \vartheta e_2 \wedge \exists w\,(e_1\ in\ w \wedge e_2\ in\ w)$.	

We may also write $e < t$ for $\vartheta e < t$ if e is an eventuality and t is a period, or $t < e$ for $t < \vartheta e$, and we can have similar abbreviations for $\subseteq$, O, $<<$ and $\lesssim$. I write $e\ at\ t$ for $\vartheta e = t$.

Note that $\exists w\,(e_1\ in\ w \wedge e_2\ in\ w)$ is now equivalent to $e_1 < e_2 \vee e_1\ \mathrm{O}\ e_2 \vee e_2 < e_1$ and to $e_1 << e_2 \vee e_2 << e_1$, so that these are three equivalent formulations of comparability. We impose three more axioms. The first says that each eventuality occurs in some possible world; the second that if e_1 and e_2 are comparable, e_2 occurs in w, and e_2's end point is not to the left of e_1's end point, then e_1 occurs in w as well; the third—slightly idealizing—says that in each world at each period of time some eventuality takes place.

AX9 $\quad \forall e\, \exists w\ \ e\ in\ w$
AX10 $\quad \forall e_1 e_2\,(e_1 << e_2 \rightarrow \forall w\,(e_2\ in\ w \rightarrow e_1\ in\ w))$
AX11 $\quad \forall t\, \forall w \exists e\ \ (e\ at\ t \wedge e\ in\ w)$

Axiom AX10 in fact says that the past is immutable: whatever has happened will always have happened. It is easy to verify that the precedence relation on eventualities is a strict partial ordering and that any two eventualities that both precede a third are comparable. That is, the following three sentences are now provable.

(35) $\forall e\ \neg e < e$
(36) $\forall e_1 e_2 e_3\, ((e_1 < e_2 \wedge e_2 < e_3) \rightarrow e_1 < e_3)$
(37) $\forall e_1 e_2 e_3\, ((e_1 < e_3 \wedge e_2 < e_3) \rightarrow (e_1 < e_2 \vee e_1 \mathrm{O}\, e_2 \vee e_2 < e_1))$

Replace 'O' in (37) by '=' and you get the usual axioms for *backwards linear orderings* or *branching time structures* (see Thomason [1970]). Of course here we don't want overlap to imply identity, since our eventualities have duration and may occur at the same time and yet be different.

7. TENSE

Bach [1983] gives a treatment of the English auxiliary within the context of categorial grammar. Bach provides categorial grammars with a feature system and assumes that tenses and aspects are functions on verb phrases (as argued for in Bach [1980]). Although we do not need the full sophistication of Bach's grammar here, we shall follow him in this last assumption and we shall use Montague's multiple slashes to encode a rudimentary feature system. In particular, revising the definitions given above, we shall write

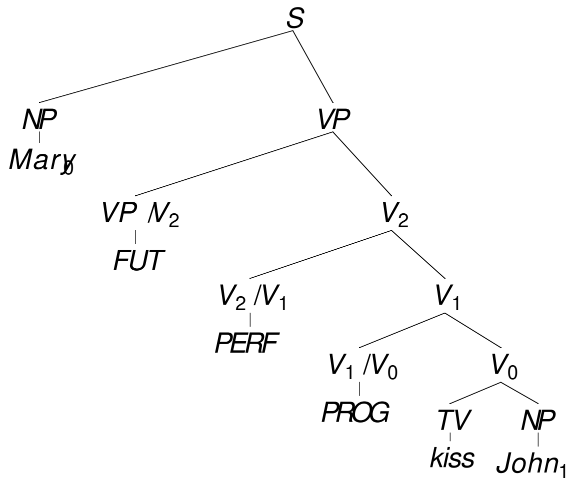
N	(common noun phrase)	for	S / E,	
V_0	(untensed verb phrase)	for	$E \setminus S$,	
V_1		for	$E \setminus^2 S$,	
V_2		for	$E \setminus^3 S$,	
VP	(tensed verb phrase)	for	$E \setminus^4 S$,	
NP	(noun phrase)	for	S / VP,	
TV	(transitive verb phrase)	for	V_0 / NP,	and
DET	(determiner)	for	NP / N.	

The idea is that V_0—V_1—V_2—VP forms a 'projection line' that provides for possibilities to hook on certain temporal operators. In particular, we shall have operators for Past, Present and Future as well as a Perfective and a Progressive operator. The following table assigns a category to each of them. Since our category-to-type rule will remain unchanged, each of these operators is interpreted as a function that takes predicates to predicates (type $[[e]e]$).

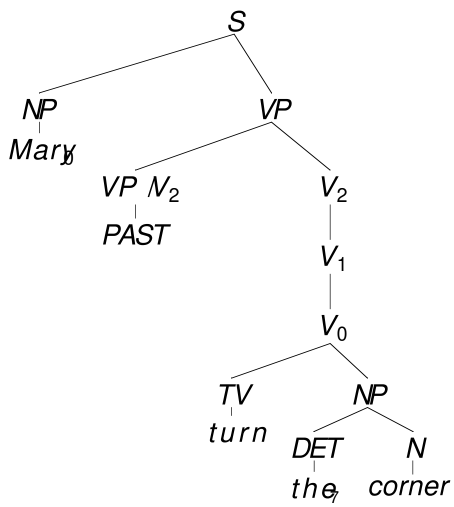
Category	*Type*	*Basic expressions*
V_1 / V_0	$[[e]e]$	*PROG*
V_2 / V_1	$[[e]e]$	*PERF*
VP / V_2	$[[e]e]$	*PRES, PAST, FUT*

These five temporal operators can now be used to bridge the gaps between V_0 and V_1, V_1 and V_2, and V_2 and VP, as in (38).

(38)



(39)



In order to allow for some operators to be skipped we add the following rule to our categorial system: any expression of category V_0 belongs to category V_1 as well, and any expression of category V_1 belongs to category V_2. As a consequence we have, for example, that (39) is generated.

It is now possible to skip *PROG* or *PERF* or both, but a choice between *PAST, PRES* and *FUT* remains obligatory. This leaves us with twelve tenses. Of course, sentences like *Mary*$_0$ *PAST turn the*$_7$ *corner* should not be left as they are and I assume some rules to convert such expressions into an acceptable form (into *Mary*$_0$ *turned the*$_7$ *corner* in this case). The same rules should ensure that (say) *John*$_0$ *FUT PERF PROG kiss she*$_4$ comes out as *John*$_0$ *will have been kissing her*$_4$.

Vendler [1967], following a tradition dating back to Aristotle, classifies predicates as states (e.g. *be drunk*), activities (*walk*), accomplishments (*build a house*), and achievements (*die*). There are many semantic and syntactic tests that help us to distinguish between these categories (see Dowty [1979]). It has been observed that verbs that express accomplishments and achievements, being *kinesis* verbs, verbs of motion, push the point of reference forward. States and activities, on the other hand, are assumed to leave the reference point as it is. Another difference is that states are interpreted as including the current point of reference, while an event expressed by a kinesis verb is either included in or (in our slightly simplified set-up) is equal to the reference point.

These differences are formalized in the translations below. As before, we let intransitive verbs and common nouns be translated as expressions of type [*e*] and transitive verbs as expressions of type [[[*e*]]*e*], but now we make a distinction between kinesis verbs such as *yawn* and *see* on the one hand and states such as *be drunk* on the other. A kinesis verb such as *yawn* tests whether the subject yawns at the current reference point (*yawn xe* intuitively stands for '*e* is an event of *x* yawning'[20]) and then assigns a new value to that reference point, setting it just after its old value. A state like *be drunk* doesn't move the reference point, it just tests whether the current reference point is included in an event of the subject's being drunk. Common nouns are treated on a par with state-like verbs ('be president', for example, is a state). We let the reference point R be a store name here; its value will always be an eventuality, so R is an expression of type $s\varepsilon$.

yawn	$\leadsto$	$\lambda x \lambda ij\,(yawn\ x\,(Ri) \wedge i\,[R]j \wedge Ri \lessapprox Rj)$
see	$\leadsto$	$\lambda Q \lambda y\,(Q \lambda x \lambda ij\,(see\ xy\,(Ri) \wedge i\,[R]j \wedge Ri \lessapprox Rj))$
be drunk	$\leadsto$	$\lambda x \lambda ij \exists e\,(i = j \wedge drunk\ xe \wedge Ri \subseteq e))$
president	$\leadsto$	$\lambda x \lambda ij \exists e\,(i = j \wedge president\ xe \wedge Ri \subseteq e))$

Here *yawn* and *be drunk* are expressions of category V_0, while *see* belongs to category TV and *president* is an N. In order to get translations of tensed verb phrases we need at least translations for the operators Past, Present and Future. I give them below. In these translations the point of speech S is a store name of type $s\tau$, while W (the current world) is a store name of type sw. Note the difference between R and S; one store contains eventualities, the other periods of time.[21]

PAST	$\leadsto$	$\lambda P \lambda x \lambda ij\,(Pxij \wedge Ri < Si \wedge Ri\ in\ Wi)$
PRES	$\leadsto$	$\lambda P \lambda x \lambda ij\,(Pxij \wedge Ri\ at\ Si \wedge Ri\ in\ Wi)$

---

[20] Note how close this brings us to a Davidsonian event semantics. I think that a theory along the lines of Parsons [1990] could easily be implemented within the present framework.

[21] This asymmetric treatment of point of speech and point of reference is chosen for technical convenience and can easily be replaced by a symmetrical setup.

FUT $\rightsquigarrow$ $\lambda P\lambda x\lambda ij\,(Pxij \wedge Si < Ri \wedge Ri \text{ in } Wi)$

As is easy to see now, the translation of *yawned,* which we get by applying the translation of the past tense to that of the untensed verb phrase *yawn* gives the term $\lambda x\lambda ij\,(yawn\ x\,(Ri) \wedge i\,[R\,]j \wedge Ri \lesssim Rj \wedge Ri < Si \wedge Ri \text{ in } Wi)$. We also find that *will be drunk* translates as $\lambda x\lambda ij\exists e\,(i = j \wedge drunk\ xe \wedge Ri \subseteq e \wedge Si < Ri \wedge Ri \text{ in } Wi)$. The past, present and future tenses each add an extra condition. The past requires that the current reference point is before the point of speech, the present requires that the reference point is at speech time and the future says that the point of reference is after speech time. In all cases the reference point is situated in the actual world. A sentence like *Mary$_0$ will be drunk,* for example, makes a statement about the actual future, not just about one of all possible futures.

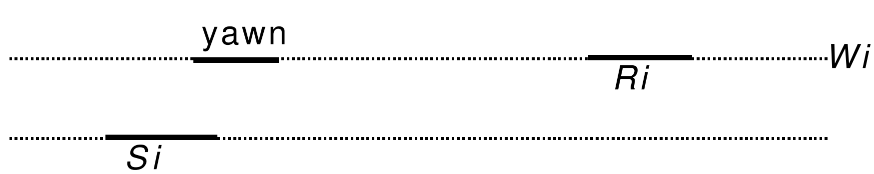
We can apply the above tense operators directly to untensed verb phrases or we may apply a perfective or progressive operator (or both) first. I translate the last two operators as shown below. The effect of *PERF* can be described as follows: first the reference point is non-deterministically set to an event that completely precedes the input reference point, then the verb is evaluated, and then the reference point is reset to its old value. The effect of *PROG* is similar: the reference point is non-deterministically set to an event that includes the input reference point, the untensed verb is evaluated, and the reference point is set back again.

PERF $\rightsquigarrow$ $\lambda P\lambda x\lambda ij\,\exists kl\,(i\,[R\,]k \wedge Rk < Ri \wedge Pxkl \wedge l\,[R\,]j \wedge Ri = Rj)$
PROG $\rightsquigarrow$ $\lambda P\lambda x\lambda ij\,\exists kl\,(i\,[R\,]k \wedge Ri \subseteq Rk \wedge Pxkl \wedge l\,[R\,]j \wedge Ri = Rj)$

The reader may wish to verify that e.g. the future perfect *will have yawned,* the result of first applying the perfective operator to *yawn* and then the future tense to the result, is now translated as in (40). We also find for example that *had been drunk* is translated as in (42). In (41) and (43) pictures are drawn that in an obvious way correspond to models for the results of applying (40) and (42) to particular *x, i* and *j.* (The lower line gives the time axis while the upper line is a possible world.)

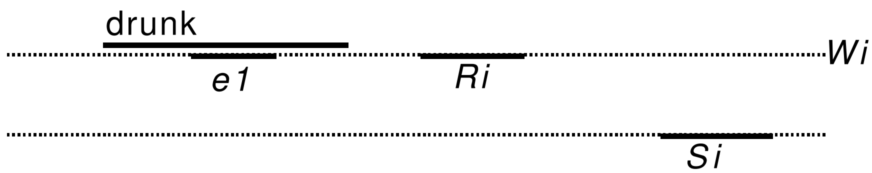
(40) *will have yawned*
$\lambda x\lambda ij\exists e\,(i = j \wedge yawn\ xe \wedge e < Ri \wedge Si < Ri \wedge Ri \text{ in } Wi)$

(41)



(42) *had been drunk*
$\lambda x\lambda ij\exists e_1 e_2\,(i = j \wedge drunk\ xe_2 \wedge e_1 \subseteq e_2 \wedge e_1 < Ri < Si \wedge Ri \text{ in } Wi)$

(43)



In a similar way we can find translations for stative and kinesis VPs in twelve tenses: Simple Past, Simple Present, Simple Future, Past Perfect, Present Perfect, Future Perfect, Continuous Past, Continuous Present, Continuous Future, and the Continuous forms of Past Perfect, Present Perfect and the Future Perfect. There is of course a clear connection between our translations and Reichenbach's temporal structures. For example, in the translation of *had been drunk* we recognize Reichenbach's *E—R—S* and *will have yawned* admits not only of the structure *S—E—R,* but also of *S, E—R* and of *E—S—R.* In Table 2 below I have given a systematic listing of the twelve tenses of the verb *yawn* that our grammar predicts, the translations it assigns to these twelve forms and—in the first six cases—the Reichenbachian forms that these translations admit.

Expression	Translation	
PRES yawn *yawns*	$\lambda x \lambda ij\ (yawn\ x\ (Ri) \wedge i\ [R]j \wedge Ri \lesssim Rj \wedge Ri\ at\ Si \wedge Ri\ in\ Wi)$	*E, R, S*
PAST yawn *yawned*	$\lambda x \lambda ij\ (yawn\ x\ (Ri) \wedge i\ [R]j \wedge Ri \lesssim Rj \wedge Ri < Si \wedge Ri\ in\ Wi)$	*E, R—S*
FUT yawn *will yawn*	$\lambda x \lambda ij\ (yawn\ x\ (Ri) \wedge i\ [R]j \wedge Si < Ri \lesssim Rj \wedge Ri\ in\ Wi)$	*S—E, R*
PRES PERF yawn *has yawned*	$\lambda x \lambda ij \exists e\ (i = j \wedge yawn\ xe \wedge e < Ri \wedge Ri\ at\ Si \wedge Ri\ in\ Wi)$	*E—R, S*
PAST PERF yawn *had yawned*	$\lambda x \lambda ij \exists e\ (i = j \wedge yawn\ xe \wedge e < Ri < Si \wedge Ri\ in\ Wi)$	*E—R—S*
FUT PERF yawn *will have yawned*	$\lambda x \lambda ij \exists e\ (i = j \wedge yawn\ xe \wedge e < Ri \wedge Si < Ri \wedge Ri\ in\ Wi)$	*E—S—R* *S, E—R* *S—E—R*
PRES PROG yawn *is yawning*	$\lambda x \lambda ij \exists e\ (i = j \wedge yawn\ xe \wedge Ri \subseteq e \wedge Ri\ at\ Si \wedge Ri\ in\ Wi)$	
PAST PROG yawn *was yawning*	$\lambda x \lambda ij \exists e\ (i = j \wedge yawn\ xe \wedge Ri \subseteq e \wedge Ri < Si \wedge Ri\ in\ Wi)$	

FUT PROG yawn *will be yawning*	$\lambda x \lambda ij \exists e\, (i = j \wedge yawn\ xe \wedge Si < Ri \subseteq e \wedge Ri\ in\ Wi\,)$
PRES PERF PROG yawn *has been yawning*	$\lambda x \lambda ij \exists e_1 e_2\, (i = j \wedge yawn\ xe_2 \wedge e_1 \subseteq e_2 \wedge e_1 < Ri \wedge Ri\ at\ Si \wedge Ri\ in\ Wi\,)$
PAST PERF PROG yawn *had been yawning*	$\lambda x \lambda ij \exists e_1 e_2\, (i = j \wedge yawn\ xe_2 \wedge e_1 \subseteq e_2 \wedge e_1 < Ri < Si \wedge Ri\ in\ Wi\,)$
FUT PERF PROG yawn *will have been yawning*	$\lambda x \lambda ij \exists e_1 e_2\, (i = j \wedge yawn\ xe_2 \wedge e_1 \subseteq e_2 \wedge e_1 < Ri \wedge Si < Ri \wedge Ri\ in\ Wi\,)$

TABLE 2.

Clearly, on this account the crudest forms of the so-called Imperfective Paradox (see Dowty [1979]) are avoided. From *I am crossing the$_2$ street,* for example, the future perfect *I shall have crossed the$_2$ street* will not follow.[22]

Note that in our approach we have reached a true synthesis between a referential and a quantificational approach to the English tenses. On the one hand we have translated the expressions given in Table 2 to closed terms of ordinary (quantificational) type logic, on the other hand Reichenbach's point of reference is clearly recognizable. We also see that although the Progressive and the Perfective operators are treated in terms of movement of the point of reference, the effect is nevertheless one of quantification over events.

A second point to note is that our treatment did not require an alteration of the category-to-type rule. In classical Montague Semantics the transition from a purely extensional fragment to a fragment in which intensions and times are taken into account demands a complication of the total set-up. In our approach this is no longer necessary. Since a state can be thought of as a *list* of items, we can simply add more items to the list if we wish to treat more complicated fragments of the language. We can thus proceed in a more or less modular way, extending our treatment of English by adding more parameters where necessary, while retaining previous analyses. For example, the present treatment of tenses is compatible with our previous account of the semantic behaviour of nominal anaphora and we can accept all translations given in section 5 except those for verbs and common nouns.

It is also possible to extend our fragment with a treatment of intensional phenomena without having to revise it. For example, we could let the following term be a translation for the auxiliary *may* of category $(NP \setminus S)/V_2$.

---

[22] Logical consequence is inclusion of denotations. A sentence A follows from a sentence B iff in all models the relation that is the denotation of the translation of B is contained in the relation that is the denotation of the translation of A. An alternative would be the DRT notion of consequence where entailment is inclusion of contents.

may $\rightsquigarrow$ $\lambda P \lambda Q \lambda ij \exists kh\ (i = j \wedge i\ [R, W]k \wedge QPkh \wedge Rk\ in\ Wk \wedge Rk\ at\ Sk)$

Some reasoning shows that, given this translation, e.g. *A_3 unicorn may be yawning* is rendered as in (44).

(44) *A_3 unicorn may be yawning*
$\lambda ij \exists e_1 e_2 e_3 \exists x\ (i = j \wedge unicorn\ xe_1 \wedge yawn\ xe_2 \wedge e_3 \subseteq e_1 \wedge e_3 \subseteq e_2 \wedge e_3\ at\ Si)$

This is the (preferred) *de dicto* reading of the sentence, the reading that does not entail the existence of a unicorn in the actual world and that does not license a pronoun picking up the referent created by the indefinite noun phrase. In order to get it we had to assign a 'lifted' category to the auxiliary *may.* To get a *de re* reading we may use one of the usual scoping mechanisms.[23]

Let's see how our little fragment models the *consecutio temporum* that is typical for linear narrative. In (45)-(49) some sentences are given from which we shall form short texts; the translations are given as well. In order to facilitate comparison with Partee [1984], the first three sentences are taken from this paper.

(45) *$Mary_1$ turned the_2 corner*
$\lambda ij \exists e\ (v_1 i = mary \wedge corner\ (v_2 i)e \wedge Ri \subseteq e \wedge i\ [R]j \wedge turn\ (v_2 i)(v_1 i)(Ri) \wedge$
$Ri \lesssim Rj \wedge Ri < Si \wedge Ri\ in\ Wi)$

(46) *$John_3$ saw her_1*
$\lambda ij\ (v_3 i = john \wedge i\ [R]j \wedge see\ (v_1 i)(v_3 i)(Ri) \wedge Ri \lesssim Rj \wedge Ri < Si \wedge Ri\ in\ Wi)$

(47) *She_1 crossed the_4 street*
$\lambda ij \exists e\ (street\ (v_4 i)e \wedge Ri \subseteq e \wedge i\ [R]j \wedge cross\ (v_4 i)(v_1 i)(Ri) \wedge Ri \lesssim Rj \wedge$
$Ri < Si \wedge Ri\ in\ Wi)$

(48) *She_1 was drunk*
$\lambda ij \exists e\ (i = j \wedge drunk\ (v_1 i)e \wedge Ri \subseteq e \wedge Ri < Si \wedge Ri\ in\ Wi)$

(49) *$John_3$ had seen her_1*
$\lambda ij \exists e\ (i = j \wedge v_3 i = john \wedge e < Ri < Si \wedge see\ (v_1 i)(v_3 i)e \wedge Ri\ in\ Wi)$

---

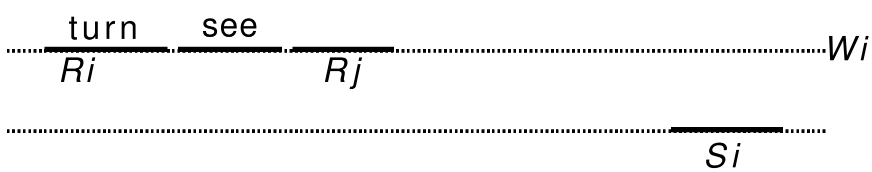
23 It may be that some of the temporal operators discussed above need similarly 'lifted' translations.

In (50) and (51) it is shown what happens if two kinesis sentences like the ones given in (45) and (46) are chained. The first event (the turning) occurs at the original reference time. Then the reference time is pushed forward to a place just after its original position; this new reference time is the time of the second event (the seeing).[24] After this the reference time is moved again, waiting for more events to come. In (52) the little text is indeed continued with one more kinesis sentence, and the process goes on: the new event is interpreted at the now current reference time and the reference time is pushed forward again. The result can be pictured as in (53).[25]

(50) *Mary$_1$ turned the$_2$ corner. John$_3$ saw her$_1$*

$\lambda ij \exists e_1 e_2\, (v_1 i = mary \wedge corner\,(v_2 i)e_1 \wedge Ri \subseteq e_1 \wedge turn\,(v_2 i)(v_1 i)(Ri) \wedge i\,[R]j \wedge v_3 i = john \wedge see\,(v_1 i)(v_3 i)e_2 \wedge Ri \lesssim e_2 \lesssim Rj \wedge e_2 < Si \wedge e_2\ in\ Wi)$

(51)



(52) *Mary$_1$ turned the$_2$ corner. John$_3$ saw her$_1$. She$_1$ crossed the$_4$ street*

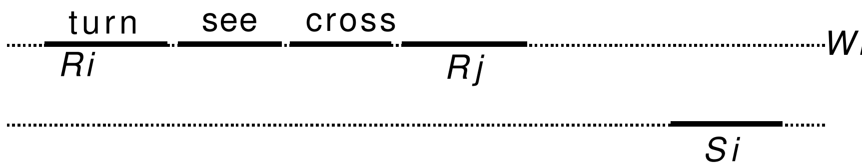
$\lambda ij \exists e_1 e_2 e_3 e_4\, (v_1 i = mary \wedge corner\,(v_2 i)e_1 \wedge Ri \subseteq e_1 \wedge turn\,(v_2 i)(v_1 i)(Ri) \wedge i\,[R]j \wedge v_3 i = john \wedge see\,(v_1 i)(v_3 i)e_2 \wedge street\,(v_4 i)e_3 \wedge e_4 \subseteq e_3 \wedge cross\,(v_4 i)(v_1 i)e_4 \wedge Ri \lesssim e_2 \lesssim e_4 \lesssim Rj \wedge e_4 < Si \wedge e_4\ in\ Wi)$

---

24 The relation 'just after', $\lesssim$, was defined to hold between two events iff the period of time associated with the first event immediately precedes the period associated with the second event. The result here is that the seeing must immediately follow the turning and that in (52) below the crossing must be immediately after the seeing. In general our account is too strict, since on the ordinary interpretation there might be short lapses of time between turning and seeing and between seeing and crossing. Natural language is vaguer than our theory predicts it to be. For the moment however, we may be content with the precise relation as a first approximation to the vague one. See Partee [1984] for a suggestion of Ewan Klein that $e_1 \lesssim e_2$ be interpreted as 'e_2 is after e_1 and there is no contextually relevant e_3 between e_1 and e_2'.

25 In these pictures I do not draw those states that are connected to the interpretations of common nouns. The output reference time *Rj* will in all cases be drawn as an eventuality occurring in *Wi* even if this is not enforced by the translation of the text. The reason is that other choices for *Rj* will be ruled out by continuations of the text. For example, in (50) the output reference time *Rj* need not be in *Wi,* but in (52) the crossing must be.

(53)

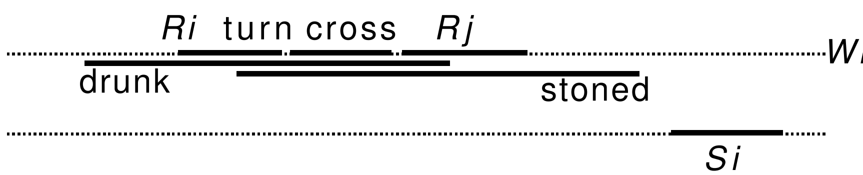


Text (54) shows what happens if a series of kinesis sentences is interrupted by one or more states. The first event after the interruption is interpreted to have occurred just after the last event before the interruption. The states all include the second event. Thus Mary's turning the corner is just before her crossing the street, but her being drunk and her being stoned take place at periods that include at least the time of her crossing. A possible model is given in (55).

(54) *Mary$_1$ turned the$_2$ corner. She$_1$ was drunk. She$_1$ was stoned. She$_1$ crossed the$_4$ street*

$\lambda ij\exists e_1e_2e_3e_4e_5(v_1 i = mary \wedge corner(v_2 i)e_1 \wedge Ri \subseteq e_1 \wedge turn(v_2 i)(v_1 i)(Ri) \wedge i[R]j \wedge street(v_4 i)e_4 \wedge e_5 \subseteq e_4 \wedge drunk(v_1 i)e_2 \wedge e_5 \subseteq e_2 \wedge stoned(v_1 i)e_3 \wedge e_5 \subseteq e_3 \wedge cross(v_4 i)(v_1 i)e_5 \wedge Ri \lesssim e_5 \lesssim Rj \wedge e_5 < Si \wedge e_5 \; in \; Wi)$

(55)

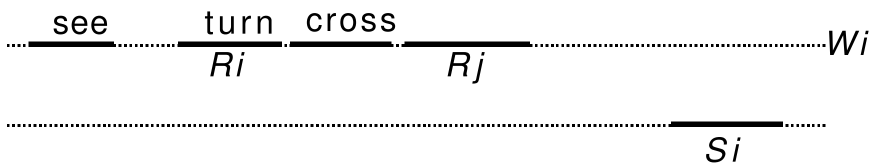


Something similar happens when a consecution of kinesis sentences is interrupted by one or more perfective sentences. In (56) the crossing is again just after the turning, but the seeing is interpreted as having occurred at some time before the crossing. (57) suggests a model.

(56) *Mary$_1$ turned the$_2$ corner. John$_3$ had seen her$_1$. She$_1$ crossed the$_4$ street*

$\lambda ij\exists e_1e_2e_3e_4(v_1 i = mary \wedge corner(v_2 i)e_1 \wedge Ri \subseteq e_1 \wedge turn(v_2 i)(v_1 i)(Ri) \wedge i[R]j \wedge v_3 i = john \wedge see(v_1 i)(v_3 i)e_2 \wedge e_2 < e_4 \wedge street(v_4 i)e_3 \wedge e_4 \subseteq e_3 \wedge cross(v_4 i)(v_1 i)e_4 \wedge Ri \lesssim e_4 \lesssim Rj \wedge e_4 < Si \wedge e_4 \; in \; Wi)$

(57)



Of course, there are other ways to connect sentences besides just sequencing them. *Temporal connectives* for instance provide alternative possibilities. Here are some possible translations for the temporal connectives *when, after* and *before.* The idea of the translation of *when* is that the reference point is first moved forward before antecedent and consequent are interpreted (in that order). The translation of *after* moves the reference point forward, then interprets the antecedent, then moves the reference point again and finishes with interpreting the consequent. *Before* is somewhat more complicated. First the reference point is moved forward to a position that need not necessarily be in the current world. The antecedent is evaluated there; but the consequent is evaluated at a new position, which is situated between the input reference point and the antecedent reference point; this new reference point is located in the current world again.

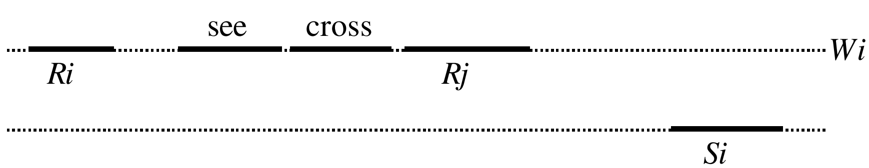
when	$\rightsquigarrow$	$\lambda pq\lambda ij\exists kl\,(i\,[R\,]k \wedge Ri < Rk \wedge pkl \wedge qlj\,)$
after	$\rightsquigarrow$	$\lambda pq\lambda ij\exists klh\,(i\,[R\,]k \wedge pkl \wedge l\,[R\,]h \wedge Ri < Rk < Rh \wedge qhj\,)$
before	$\rightsquigarrow$	$\lambda pq\lambda ij\exists klh\,(i\,[R, W\,]k \wedge i\,[R\,]h \wedge pkl \wedge qhj \wedge Ri < Rh < Rk\,)$

In (58) and (60) below we work out two example sentences and in (59) and (61) we draw pictures again that suggest possible models. As is apparent from (61) a *before* sentence does not entail its antecedent.

(58) *When John$_3$ saw her$_1$ she$_1$ crossed the$_4$ street*

$\lambda ij\exists e_1e_2e_3\,(i\,[R\,]j \wedge v_3\,i = john \wedge see\,(v_1\,i\,)(v_3\,i\,)e_1 \wedge street\,(v_4\,i\,)e_2 \wedge cross\,(v_4\,i)(v_1\,i\,)e_3 \wedge Ri < e_1 \lesssim e_3 \lesssim Rj \wedge e_3 < Si \wedge e_3 \subseteq e_2 \wedge e_3\ in\ Wi)$

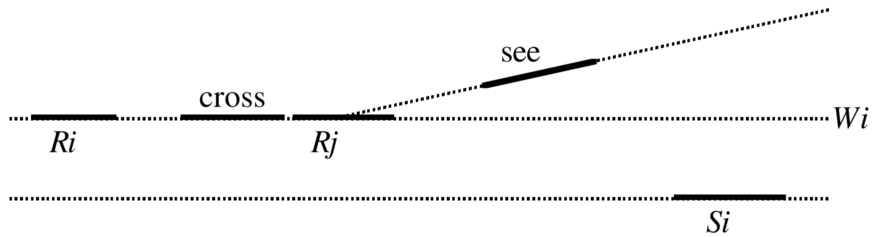
(59)



(60) *Before John$_3$ saw her$_1$ she$_1$ crossed the$_4$ street*

$\lambda ij\exists e_1e_2e_3\,(i\,[R\,]j \wedge v_3\,i = john \wedge see\,(v_1\,i\,)(v_3\,i\,)e_1 \wedge street\,(v_4\,i\,)e_2 \wedge cross\,(v_4\,i\,)(v_1\,i\,)e_3 \wedge Ri < e_3 < e_1 \wedge e_3 \lesssim Rj \wedge e_1 < Si \wedge e_3 \subseteq e_2 \wedge e_3\ in\ Wi\,)$

(61)



REFERENCES

Bach, E.: 1980, Tenses and Aspects as Functions on Verb Phrases, in Chr. Rohrer (ed.), *Time, Tense and Quantifiers,* Niemeyer, Tübingen, 19-37.

Bach, E.: 1983, Generalized Categorial Grammars and the English Auxiliary, in F. Heny and B. Richards (eds.), *Linguistic Categories: Auxiliaries and Related Puzzles, Vol. II,* Reidel, Dordrecht.

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

Bäuerle, R., Egli, U., and Von Stechow, A. (eds.): 1979, *Semantics from Different Points of View,* Springer, Berlin.

Van Benthem, J.F.A.K.: 1983, *The Logic of Time,* Kluwer, Dordrecht (2nd edition 1991).

Van Benthem, J.F.A.K.: 1989, Semantic Parallels, in H.D. Ebbinghaus et al. (eds.), *Logic Colloquium. Granada 1987,* North-Holland, Amsterdam, 331-375.

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

Dowty, D.R.: 1979, *Word Meaning and Montague Grammar,* Reidel, Dordrecht.

Van Eyck, J.: 1991, Presupposition Failure: A Comedy of Errors, manuscript.

Friedman, J. and Warren, D.: 1980, Lambda Normal Forms in an Intensional Logic for English, *Studia Logica* **39**, 311-324.

Gabbay, D. and Günthner, F. (eds.): 1983, *Handbook of Philosophical Logic*, Reidel, Dordrecht.

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

Goldblatt, R.: 1987, *Logics of Time and Computation,* CSLI Lecture Notes, Stanford.

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

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

Harel, D.: 1984, Dynamic Logic, in Gabbay & Günthner [1983], 497-604.

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

Hinrichs, E.: 1981, *Temporale Anaphora im Englischen,* Staatsexamen thesis, University of Tübingen.

Hinrichs, E.: 1986, Temporal Anaphora in Discourses of English, *Linguistics and Philosophy* **9**, 63-82.

Kamp, H.: 1979, Events, Instants and Temporal Reference, in Bäuerle, Egli & Von Stechow [1979], 376-417.

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.

Kamp, H. and Reyle, U.: to appear, *From Discourse to Logic,* manuscript.

Kamp, H. and Rohrer, Chr.: 1983, Tense in Texts, in R. Bäuerle, Chr. Schwarze and A. Von Stechow, *Meaning, Use and Interpretation of Language,* Berlin.

Lewis, D.: 1979, Score Keeping in a Language Game, in Bäuerle, Egli & Von Stechow [1979], 172-187.

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

Muskens, R.A.: 1989a, Going Partial in Montague Grammar, in R. Bartsch, J.F.A.K. van Benthem and P. van Emde Boas (eds.), *Semantics and Contextual Expression,* Foris, Dordrecht, 175-220.

Muskens, R.A.: 1989b, *Meaning and Partiality,* Dissertation, University of Amsterdam.

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

Oversteegen, E.: 1989, *Tracking Time,* Dissertation, University of Amsterdam.

Parsons, T.: 1990, *Events in the Semantics of English: a Study in Subatomic Semantics,* MIT Press, Cambridge, Massachusetts.

Partee, B.H.: 1984, Nominal and Temporal Anaphora, *Linguistics and Philosophy* **7**, 243-286.

Pratt, V.R.: Semantical Considerations on Floyd-Hoare Logic, *Proc. 17th IEEE Symp. on Foundations of Computer Science,* 109-121.

Reichenbach, H.: 1947, *Elements of Symbolic Logic,* Macmillan, New York.

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

Thomason, R.H.: 1970, Indeterminist Time and Truth Value Gaps, *Theoria* **36**, 264-281.

Vendler, Z.: 1967, *Linguistics in Philosophy,* Cornell University Press, Ithaca.