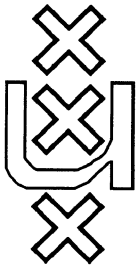
Institute for Language, Logic and Information

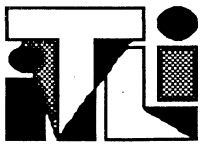
THE FINE-STRUCTURE OF CATEGORIAL SEMANTICS

Johan van Benthem

ITLI Prepublication Series
for Logic, Semantics and Philosophy of Language LP-89-01



University of Amsterdam



Institute for Language, Logic and Information
Instituut voor Taal, Logica en Informatie

THE FINE-STRUCTURE OF CATEGORIAL SEMANTICS

Johan van Benthem
Department of Mathematics and Computer Science
University of Amsterdam

Received January 1989

Correspondence to:

Faculteit der Wiskunde en Informatica
(Department of Mathematics and Computer Science)
Roetersstraat 15
1018WB Amsterdam

or

Faculteit der Wijsbegeerte
(Department of Philosophy)
Grimburgwal 10
1012GA Amsterdam

THE FINE-STRUCTURE OF CATEGORIAL SEMANTICS

revised version
december 1988

Johan van Benthem

Institute for Language, Logic and Information
University of Amsterdam

Contents

1. Introduction

In the linguistic study of syntax, various formalisms have been developed for measuring the combinatorial power of natural languages. In particular, there is a well-known *hierarchy of grammars* which can be employed to calibrate syntactic complexity. No similar tradition exists in linguistic semantics, however. Although there exist powerful formalisms for stating truth conditions, these are usually presented as 'monoliths': without any obvious mode of 'fine-tuning'. The purpose of this paper is to show how semantics has its fine-structure too, when viewed from a proper logical angle.

The framework used to demonstrate this correspondence is that of Categorial Grammar. In this field, a syntactic hierarchy is provided by the landscape of logical *calculi of implication*, starting from an Ajdukiewicz type system with Modus Ponens only and then ascending up to a full intuitionistic conditional logic, which also allows conditional subproofs. These correspond to various 'engines' for driving categorial combination, which can be studied as to their mathematical linguistic properties by employing the tools of logical Proof Theory. (See Klein & van Benthem, eds., 1988, Buszkowski, Marciszewski & van Benthem, eds., 1988 or Oehrle, Bach & Wheeler, eds., 1988 for a fuller account of this paradigm.)

Being logical calculi of deduction, these various systems of categorial grammar also come with a systematic *semantics*. Each categorial derivation corresponds effectively with a type-theoretic term which encodes the corresponding semantic reading (see van Benthem 1986, chapter 7). This correspondence again reveals a landscape, this time of different *fragments* of a full type-theoretic language employing function application and lambda abstraction, i.e., a standard medium of expression in model-theoretic semantics. In fact, what we have then is a *semantic hierarchy* of richer or poorer vehicles for expressing denotations of linguistic expressions, which may be studied by employing the tools of logical Model Theory. If Montagovian Compositionality is the thesis that Type Theory provides enough 'abstract glue' for implementing Frege's Principle for natural language, the issue is now to determine just how much glue is needed from case to case. (This theme is developed in van Benthem 1986, 1987b.)

The further organization of this paper is as follows. Section 2 contains a presentation of standard type-theoretical semantics, emphasizing the general semantic questions about natural language engendered by it. The fine-structure perspective is then elaborated in Section 3. Next, in Section 4, some variations on the standard approach are reviewed, in order to show how the concerns and methods so far transfer to more complex modelings of linguistic phenomena. Finally, some more

technical background material has been collected into the appendices of Section 5.

2. Type-Theoretical Semantics

2.1 Preliminaries

We shall be using a standard type theory, based on primitive types

e ('entity') and t ('truth value') ,

with at least two rules for forming complex types:

(a, b) (functions from a-type to b-type objects)

a.b (ordered pairs of a-type and b-type objects)

These refer to semantic structures built up as follows ('function hierarchies')

D_e is some initial universe of individuals
D_t is the truth value domain {0,1}
$D_{(a,b)} = \{f \mid f: D_a \rightarrow D_b\}$
$D_{a.b} = D_a \times D_b$.

A suitable type-theoretic companion language then has variables and constants of each type, as well as (at least) the following ways of forming complex terms:

Application:	if A is a term of type (a, b) and B a term of type a , then A(B) is a term of type b
Abstraction:	if A is a term of type b , and x a variable of type a , then $\lambda x.A$ is a term of type (a, b)
Pairing:	if A is a term of type a , and B a term of type b , then <A,B> is a term of type a.b
Projection:	if A is a term of type a.b , then $\pi_L(A)$, $\pi_R(A)$ are terms of types a, b .

Interpretation of this language in the above structures is standard.

Moreover, there are various possible extensions to its resources, such as adding an *identity* predicate = in arbitrary types with its proper meaning.

Remark: Product types and the attendant terms are not very prominent in what follows. They have been added, however, in order to demonstrate that the type-theoretical approach is not confined to working with functions and function types only. ♠

Recently, various semantic modifications of this framework have been proposed, including 'many-sorted' prunings of overcrowded function hierarchies, as well as more partial, 'information-oriented' versions of the latter. But, the standard format adopted here will suffice for illustrating the issues that we are concerned with.

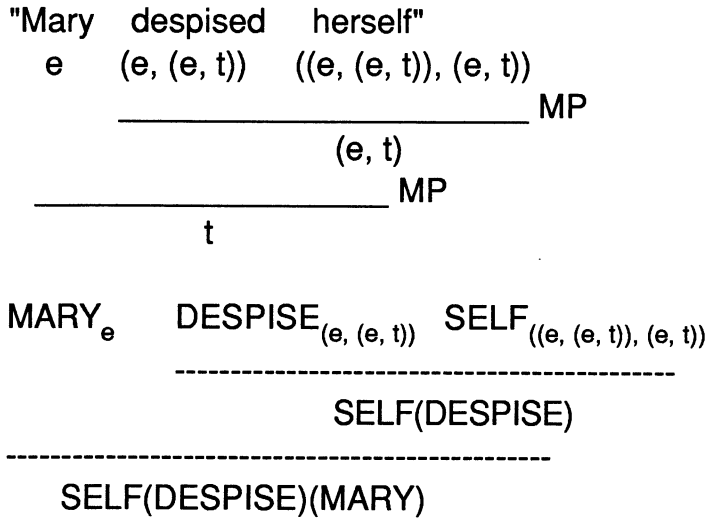
Now, *terms* in this type-theoretic language may be assigned systematically to *derivations* in Categorial Grammar. This may be illustrated as follows (for a full exposition, see van Benthem 1986, 1987b):

Example: Reflexives.

- As has often been observed, reflexives "self" (and even bound pronouns) may be viewed as argument reducers on relations : i.e., as items in type

((e, (e, t)), (e, t)) ("[despise] oneself") .

Here is a categorial derivation exemplifying this, together with its type-theoretic meaning:



Note how Function Applications reflect occurrences of Modus Ponens.

By invoking the obvious denotation of the reflexive operator, being

$$\lambda R_{(e,\,(e,\,t))}.\ \lambda y_e.\ R(y)(y)\ ,$$

the latter formula may be converted into the intended meaning

DESPISE(MARY)(MARY).

- But also, reflexives may occur in parametrized cases, such as ("[teach] oneself [a lesson]"), where the relevant type has become

$$((e,\,(e,\,(e,\,t))),\,(e,\,(e,\,t)))\ .$$

Here, a categorial derivation producing the intended meaning will be more complex, invoking a stronger calculus of type implication:
(where TEACH(x)(y)(z) stands for z's teaching x to y)

$$\frac{\dfrac{\dfrac{1}{e}\quad (e,\,(e,\,(e,\,t)))}{(e,\,(e,\,t))}\ \text{MP}\qquad ((e,\,(e,\,t)),\,(e,\,t))}{(e,\,t)}\ \text{MP}$$

$$\frac{(e,\,t)}{(e,\,(e,\,t))}\ \text{COND, withdrawing assumption 1}$$

$$\frac{x_e\qquad \text{TEACH}_{(e,\,(e,\,(e,\,t)))}}{\text{TEACH}(x)}\qquad \text{SELF}_{((e,\,(e,\,t)),\,(e,\,t))}$$

$$\frac{\text{SELF}(\text{TEACH}(x))}{\lambda x_e.\ \text{SELF}(\text{TEACH}(x))}$$

Note how, this time, lambda abstraction encodes the final Conditionalization step in the categorial derivation.

Again, the ordinary definition of reflexives may be plugged in to obtain the desired meaning

$$\lambda x_e.\ \lambda y_e.\ \text{TEACH}_{(e,\,(e,\,(e,\,t)))}(x)(y)(y)\ .$$

- And finally, even reflexivization in prepositional phrases, superficially quite different, exhibits the same pattern semantically:

"to"	plus	"oneself"
(e, ((e, t), (e, t)))		((e, (e, t)), (e, t))

may be combined into a reflexive prepositional phrase with a suitable meaning in type ((e, t), (e, t)) :

"to oneself" : $\lambda P_{(e,t)}.\ \lambda x_e.\ TO(x)(P)(x).$

Here is the relevant calculation:

$$\dfrac{\dfrac{2 \qquad \dfrac{1 \quad e \qquad (e, ((e, t), (e, t)))}{(e, t), (e, t)}\ MP}{(e, t)}\ MP \qquad \dfrac{\dfrac{(e, t)}{(e, (e, t))}\ COND,\ \text{withdrawing 1} \qquad ((e, (e, t)), (e, t))}{(e, t)}\ MP}{((e, t), (e, t))}\ COND,\ \text{withdrawing 2}$$

$$\dfrac{\dfrac{\dfrac{P_{(e, t)} \qquad \dfrac{x_e \qquad TO_{(e, ((e, t), (e, t)))}}{TO(x)}}{TO(x)(P)}}{\lambda x_e.\ TO(x)(P)}}{REF(\lambda x_e.\ TO(x)(P))}$$

$$\lambda P_{(e, t)}.\ REF(\lambda x_e.\ TO(x)(P)))$$

which reduces to $\lambda P.\ \lambda x.\ TO(x)(P)(x)$ by the definition of REF. ♠

What this example also demonstrates is an important facility. A type-theoretic framework is useful in semantics because it supports a search for maximal *generality* for phenomena which we have discovered, across various linguistic categories. Reflexivization is one such wide-ranging phenomenon; further illustrations (involving, e.g., Boolean structure) will be found below.

In the following parts of this Section, we shall consider some of these general themes, including their logical background theory. It may be of interest to compare the selection of topics made here with that in Gallin 1975, which, although otherwise quite useful, still represents the standard logical theory of the Montagovian paradigm, without much linguistic input.

2.2 Derivations and Readings

As the preceding discussion has shown, a derivation of some type b from a sequence of types A encodes a meaning which may also be expressed in a lambda form

$$\tau_b [\{ x_a \mid a \in A \}] \ ,$$

where the term τ of type b has parameters x_a occurring for each $a \in A$. Note that the translation is not exclusively 'type-driven' (to use a modern slogan): to one derivable transition

$$A \Rightarrow b \ ,$$

there may correspond different derivations, and hence possibly different readings. Thus, it is the derivation which is the unit of linguistic meaning, rather than the surface string. On the other hand, not every difference in derivational structure must cause a detectable difference in semantic meaning. Here are some illustrations of what can happen in general.

Example: Varying Numbers of Readings.

(1) $e\ (e, t) \Rightarrow t$.

Here, there is only one meaning, expressible in the form

$x_{(e, t)}(y_e)$.

To be sure, different categorial derivations do exist here, such as

$$\frac{e \quad (e,t)}{t}\ MP$$

versus

$$\cfrac{\cfrac{e \quad \overset{1}{(e, t)}}{t}\ MP}{((e, t), t)}\ COND,\ withdrawing\ 1 \qquad (e, t)$$
$$\text{MP}$$
$$t$$

But, the more complex one can be *reduced* to the simpler one by means of 'normalization': a proof-theoretic technique which does not affect meaning.

(2) $e \Rightarrow ((e, e), e)$.

Here, different meanings exist, witness the non-equivalent derivations

$$\cfrac{\cfrac{e \quad \overset{1}{(e, e)}}{e}\ MP}{((e, e), e)}\ COND,\ withdrawing\ 1 \quad : \quad \lambda x_{(e, e)} \cdot x_{(e, e)}(y_e)$$

$$\cfrac{\cfrac{e \quad \overset{1}{(e, e)}}{e}\ MP \quad \overset{1}{(e, e)}}{e}\ COND,\ withdrawing\ 1 \quad : \quad \lambda x_{(e, e)} \cdot x_{(e, e)}\,(x_{(e, e)}(y_e))\ ;$$

etcetera.

(3) $(t, t)\ t\ (t, t) \Rightarrow t$.

This case is ambiguous too, leading to different scope orders:

$x_{(t, t)}(z_{(t, t)}(y_t))$ versus $z_{(t, t)}(x_{(t, t)}(y_t))$.

(4) In fact, cases with exactly n non-equivalent readings, for each finite n , may be obtained from the following valid transition:

e, ...(n times)... , e $\Rightarrow$ e .

(5) Finally, more complex examples may be found in van Benthem 1987c, 1988b, concerning the possible ways of construing transitive sentences with complex NPs:

$((e, t), t)\ \ (e, (e, t))\ \ ((e, t), t) \Rightarrow t$;

or deriving binary determiners from underlying unary quantifiers. ♠

This observed diversity of readings for linguistic expressions suggests various more systematic questions. One is *how many readings* exist for given type transitions. No algorithm is known for computing this number in general - but still, some facts can be noted. For instance, Statman 1980 characterizes precisely those derivable transitions in a typed lambda calculus with one basic type which have finitely many readings. In our case, there are more primitive types than just one: which may also be subject to additional restrictions (see below). But, Statman's result can probably be extended.

Moreover, special cases are of interest: in particular, the 'unambiguous' type transitions having just one reading - or in other words, those logical laws which have essentially just *one proof*.

Example: Unambiguous Transitions.
Well-known examples of this kind are such 'type change laws' as the Montague Rule for raising denotations

$a \Rightarrow ((a, b), b)$,

whose only corresponding lambda normal form may be seen to be (by a simple syntactic calculation)

$$\lambda x_{(a,\,b)}.\ x(u_a)\ .$$

Likewise, the so-called Geach Rule for composing denotations

$$(a, b)\quad (b, c)\ \Rightarrow\ (a, c)$$

has essentially just the lambda normal form

$$\lambda x_a.\ u_{(b,\,c)}(v_{(a,\,b)}(x))\ .$$

Thus, for important parts of current 'flexible categorial grammar', the difference between derivations and mere transitions on type strings turns out to be inessential after all. ♠

How can it be recognized whether one of these cases occurs? Here, we can only offer a

Conjecture: The question whether a given type transition has exactly n type-theoretic readings, where n is finite or 'infinity', is decidable.

Behind all this lies a perhaps more important semantic question concerning the actual *enumeration* of all possible readings for a given transition. What we need is some *systematic method* for accomplishing this. One such method, based on formal grammars and automata, is presented in Appendix 5.1. Here is a sample outcome (cf. van Benthem 1987c).

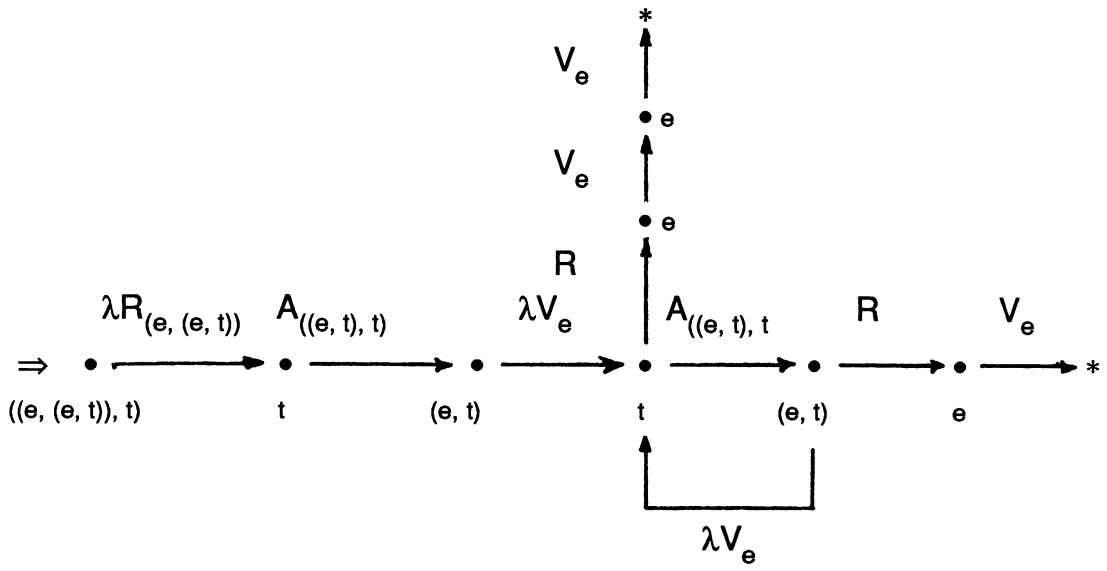
Example: Transitive Sentences with Complex Subjects and Objects. Obviously, there are the usual scope possibilities with the transitive verb or its passive form:

NP1 (λx. *NP2* (*TV* (x)))	('every boy loves a girl')
NP2 (λx. *NP1* (*TV* (x)))	('a girl loves every boy')
NP1 (λx. *NP2* (λy. *TV* (y)(x)))	('every boy is loved by a girl')
NP2 (λx. *NP1* (λy. *TV* (y)(x)))	('a girl is loved by every boy')

But in general, further cases arise, such as the reflexive form

NP1 ($\lambda x.$ *TV* (x)(x)) .

In fact, all possibilities are generated by the following finite state automaton (see Appendix 5.1 for further information):



Note that traversing this automaton will produce an infinity of possible non-equivalent lambda normal forms for the above transition. ♠

This example also introduces a further subtlety to the issue of semantic readings. For, although the number of possible readings for the transitive sentence pattern is infinite in a global sense, 'locally' inside each model, there will only be *finitely* many distinct possibilities. The reason lies in the special structure of the truth value domain D_t , which was supposed to be finite, indeed two-valued. By an additional argument (cf. again Appendix 5.1, or van Benthem 1987c), it may be shown that:

Given any two items A, B in the domain $D_{((e, t), t)}$ of some model, the only items in $D_{((e, (e, t)), t)}$ application/lambda definable from these are those generated by forms $\lambda R.$ followed by a matrix in the following list (with 'N' standing for A or B , and '(¬)' indicating an optional negation):

(¬)N ($\lambda x_e.$ (¬)R(x)(x))

(¬)N ($\lambda x_e.$ (¬)N ((¬)R(x)))

(¬)N ($\lambda x_e.$ (¬)N ($\lambda y_e.$ (¬)R(y)(x))) .

Cases such as these may be called 'locally finite'.

Now, we turn to some further aspects of admissible readings, focussing on what may be called 'constraints on truth conditions'.

A First Encounter with Occurrences

The preceding discussion has still missed some subtleties concerning expressibility of 'readings' for linguistic expressions. We are being given an expression with, say, n components of types $a_1, \ldots, a_n$, and the question is if and how these may be composed into a meaning of type b. Now, this statement of the problem already seems to convey some restrictions on type-theoretic forms $\tau_b[\{a_1, \ldots, a_n\}]$ which are admissible here. For instance, it seems reasonable to demand that

> each type a_i have at least one corresponding parameter u_{ai} occur in τ.

But then, already one of the earlier examples would be ruled out, namely the n readings supposedly corresponding to the valid transition 'n times e to e'. For, the relevant terms here would be single u_{ai}. (Incidentally, using *product types*, the requirement could be met trivially, by taking some suitable projection out of the ordered sequence of all u_{ai}.)

So, conditions on the *format of truth conditions* come to the fore, in particular concerning *occurrences* of terms within these. Here is another example. Intuitively, it would seem that the parameters u_{ai} standing for the component expressions can be used just once, as happens in actual sentence construction. But then, there is another demand on our schema τ:

> the free parameters u_{ai} should occur *only once* in τ.

Again, this has repercussions for the earlier examples. For instance, in the given schema for transitive sentences, only finitely many readings remain: as only two N positions can be filled. (But, e.g., there is still an infinity of admissible readings for the earlier transition $e \Rightarrow ((e, e), e)$.)

But, it would seem that this new requirement too, can be circumvented quite easily. For, if τ contains two occurrences of some parameter u, then we can rewrite it to the equivalent term $\lambda x.\ [x/u]\tau\ (u)$, which obeys the letter of the above requirement. But

evidently, this move violates its spirit: and hence, we shall be led to consider lambda terms having certain restrictions on lambda binding throughout, in Section 3 below. In other words, we have encountered a reasonable source of fine-structure for truth conditions.

Disregarding Occurrences

Next, there are also opposite cases, where an emphasis on occurrences seems quite inappropriate. For instance, in the setting of some specific model, there are natural questions of *definability* from given items:

> Suppose that objects $u_1, \ldots, u_n$ are given, in domains of types $a_1, \ldots, a_n$. Which objects in some target domain D_b are lambda definable from these?

In this case, unlike the previous one, we are not concerned with any particular piece of syntax awaiting interpretation. And in fact, no constraints on the occurrences of parameters in definitions for these objects seem implied. Therefore, the question can be of high complexity. For instance, given some function u_1 on the natural numbers and some natural number u_2, the question whether some other natural number v is definable from these two is in general not decidable. On *finite* base *domains*, however, the above question may in fact be decidable. The statement that it does is (a general form of what is) known in the literature as 'Plotkin's Conjecture' (cf. Statman 1982).

This second kind of question has a more linguistic counterpart after all, be it not in terms of interpreting expressions, but rather in terms of *language recognition*. A language is a set of admissible strings over some finite alphabet, and the recognition problem is to see if an arbitrary string belongs to this set. Here, items in the alphabet correspond to the above parameters, and in principle, strings in the language can have arbitrary numbers of occurrences of these. Now, semantic problems like the above arise, once we no longer talk about recognizing strings, but objects denoted by strings. For instance, given a context-free grammar and some map from symbols in its alphabet to objects, as well as some interpretation schema for its rewrite rules, we can ask of an arbitrary object in the relevant domain whether it belongs to the obvious 'denotation class' of this grammar. (Incidentally, the language recognition problem is the special case where the interpretation map is the syntactic identity function, wih mere concatenation for the rules.) For instance, the recognition problem for denotation sets of context-free grammars is

undecidable. [The above natural numbers example shows this, with respect to the grammar $\{ N \Rightarrow u_2,\ N \Rightarrow u_1 N \}$.] But again, we have an interesting

Question: Is the semantic recognition problem decidable on all *finite* structures?

Thus, again we see how the usual concerns of mathematical linguistics may be transferred from pure syntax to semantics. ♠

Finally, whatever constraints have been considered up till now, a fact is that the above calculus produces far too many 'readings' for most ordinary linguistic expressions. For instance, out of the above four scope readings for complex transitive sentences, only two are really plausible. Also, e.g., an unambiguous expression like "Every dinosaur died" would still obtain two readings, namely the inclusion of DINOSAUR in DIED, but also its converse.

There are various strategies for coping with this *overgeneration* in Categorial Grammar. One is to impose restrictions on categorial calculi: either on their axioms and rules of inference, or by imposing some filter on admissible derivations. (For instance, derivations with an overly high 'computational cost' might be banned.) Another is to encode some more syntax into the type structure: as is done in the 'directed calculi' of Lambek/Bar-Hillel (see Moortgat 1988 for an up-to-date presentation.) But, in more semantic terms , it is quite attractive to view the type-theoretic semantics for categorial derivations as mapping out a 'logical space' of a priori denotational possibilities, out of which natural language only realizes those satisfying certain additional 'denotational constraints'. In the latter vein, Keenan 1988 has some interesting empirically motivated semantic universals constraining the logical latitude of quantifier scoping.

2.3 Denotational Constraints

In the study of various special categories of expression, one encounters so-called 'denotational constraints', being general structural properties which all, or at least, many important denotations for linguistic expressions turn out to possess. Prominent examples are found in the study of *generalized quantifiers* (cf. van Benthem 1986): where all determiners appear to be *conservative*, while many important ones are *logical* or *monotone*. Our type theory provides a convenient setting for making such notions quite general across arbitrary categories of expression.

One major source of denotational constraints is the area of *Boolean*

operators, which provides many examples of concepts with a universal meaning (cf. Keenan & Faltz 1985). For instance, there is a general relation of hereditary *inclusion* ('Boolean implication') defined as follows

$\leq_t$ is $\leq$

$\leq_e$ is $=$

$f \leq_{(a, b)} g$ if *for all* $x \in D_a$, $f(x) \leq_b g(x)$

$\langle f, g\rangle \leq_{a.b} \langle i, j\rangle$ if $f \leq_a i$ *and* $g \leq_b j$.

Now, an object $f \in D_{(a, b)}$ is *monotone* in its a-argument if

$x \leq_a y$ implies $f(x) \leq_b f(y)$, *for all* $x, y \in D_a$.

And more general versions of this notion are possible, referring to other argument types waiting inside b . This generalizes the well-known notion of monotonicity found with generalized quantifiers, expressing a certain 'inferential parallellism' between an expression and some of its parts.

Other examples of general Boolean structure concern such notions as *homomorphisms* from one Boolean type domain to another [the earlier mentioned reflexivity operator REF is one example] - or *continuous* operators, which are already computable from their atomic input, as they commute with arbitrary Boolean joins of their arguments.

A final example of Boolean structure arises with the already mentioned general constraint of *conservativity*. I.e., for determiners D , the first argument turns out to restrict the second:

D AB iff D A(B$\cap$A) .

In the process of categorial combination, this phenomenon proliferates.

Example: Conservativity in Complex Transitive Sentences.
The denotation of a sentence pattern

Q1 A R Q2 B

may be computed as follows, in accordance with the simplest categorial derivation of the type transition

((e, t), ((e, t), t)) (e, t) (e, (e, t)) ((e, t), ((e, t), t)) (e, t) $\Rightarrow$ t :

$Q1\ (A, \lambda x.Q2\ (B, R\ (x)))$ iff

$Q1\ (A, \lambda x.Q2\ (B, \lambda y.(R\ (x)(y) \wedge B\ (y)) \wedge A\ (x)))$ iff

$Q1\ (A, \lambda x.Q2\ (B, \lambda y.(R\ (x)(y) \wedge B\ (y) \wedge A\ (x)) \wedge A\ (x)))$ iff

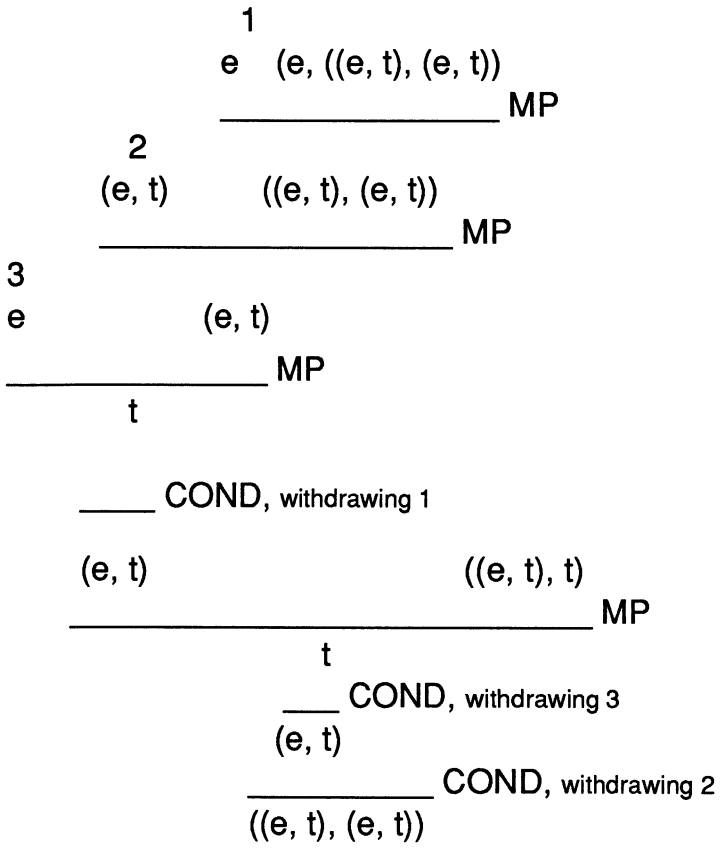
$Q1\ (A, \lambda x.Q2\ (B, \lambda y.(R\ (x)(y) \wedge B\ (y) \wedge A\ (x)))$.

I.e., Q1A R Q2B holds iff Q1A $R \cap (A \times B)$ Q2B does. ♠

And similar phenomena occur in other linguistic patterns. For instance, in evaluating an expression like "walk to every city", the common noun "city" will come to restrict the individual argument of the preposition "to", whose type may be taken to be (e, ((e, t), (e, t)) , as we did already in Section 2.1. A full calculation goes as follows.

Example: Computing Conservativity in Prepositional Phrases.

Derivation tree:



Corresponding lambda term:

$$\frac{x_e \qquad TO_{(e,\,((e,\,t),\,((e,\,t))}}{z_{(e,\,t)} \qquad TO(x)}$$

$$\frac{y_e \qquad TO(x)(z)}{TO(x)(z)(y)}$$

$$\frac{\lambda x_e.\ TO(x)(z)(y) \qquad EVERY\ CITY_{((e,\,t),\,t)}}{EVERY\ CITY(\lambda x_e.\ TO(x)(z)(y))}$$

$$\frac{EVERY\ CITY(\lambda x_e.\ TO(x)(z)(y))}{\lambda y_e.\ EVERY\ CITY(\lambda x_e.\ TO(x)(z)(y))}$$

$$\lambda z_{(e,\,t)}.\ \lambda y_e.\ EVERY\ CITY(\lambda x_e.\ TO(x)(z)(y))$$

Here, by ordinary conservativity, the last three lines may replace their part $\lambda x.\ TO(x)(z)(y)$ by $\lambda x.\ (TO(x)(z)(y) \wedge CITY(x))$. ♠

Remark. Incidentally, there are several different semantic construals of this expression in our general format. In general categorial terms, the transition (e, ((e, t), (e, t)) ((e, t), t) ⇒ ((e, t), (e, t)) also has readings such as the following (with U of type (e, ((e, t), (e, t)) and V of type ((e, t), t)) :

$$\lambda z_{(e,\,t)}.\ \lambda x_e.\ V(U(x)(z)) \ ,$$

which, when plugged into the above example, would express BEING WALKED TO BY EVERY CITY . Actually, the latter reading too would have its own appropriate form of Conservativity, computed as above. ♠

By the above mechanism of categorial evaluation, every common noun in a quantifier phrase will come to restrict at least one individual argument position, in an entirely predictable manner. By itself, this then gives us a general principle of Conservativity across the whole language. (Compare also the discussion of 'Restriction' as a feature of logical constants in van Benthem 1988b, for a connection with computation of denotations on universes of minimal size.) Again, this illustrates the

earlier point in Section 2.1 about the beneficial *generality* inherent in a type-theoretic framework.

Remark: Constraining Readings.
Conservativity provides one instance of the way in which denotational constraints may rule out type-theoretically possible readings, as suggested in Section 2.2. For instance, a sequence of the form 'Determiner, Common Noun, Verb Phrase' (say, "every sinner repented") will only be conservative with the arguments of the determiner applied in the order given. A reading in the converse order, although a priori possible, will in general violate conservativity: witness the non-conservative converse of "all", being "only". ♠

Next, for a different kind of constraint, we turn to another form of semantic behaviour.

Model-theoretically, *logicality* of an object f in some type domain D_a involves f's being insensitive to the specific nature of individuals: that is, f will be a fixed point of all *permutations* of the individual domain D_e (lifted to arbitrary domains D_a in a canonical manner). It may be shown that every closed term of a full type theory, even involving identities, defines a logical denotation in this sense - and that, at least, starting from *finite* individual domains, a converse holds too.

Example: The earlier reflexivization operator is a logical item in its type, and indeed, the only reducer of binary predicates to unary ones which is a logical homomorphism. (Cf. van Benthem 1988b, also for many further general aspects of logicality.) ♠

For the pure lambda calculus without identity, this feature can be strengthened to one found in Plotkin 1980, namely invariance under *individual relations*. Adapted to the present setting, this notion reads as follows. First, we define hereditary relations on the whole hierarchy of domains:

Let R_e be any binary relation on D_e.
Let R_t be the identity.
And inductively, let
$f\ R_{(a,b)}\ g$ iff *for all* $u, v \in D_a$: $u\ R_a\ v \rightarrow f(u)\ R_b\ g(v)$.
(The obvious clause for ordered pairs is omitted here.)

An item f in a type domain D_a is *relation-invariant* if

$f\ R_a\ f$, for all individual relations R .

Now, a straightforward induction shows that, if τ is any term with free variables $x_1, \ldots, x_n$, then for all sequences $a_1, \ldots, a_n, b_1, \ldots, b_n$ of the appropriate types:

$$\text{if } a_i\ R\ b_i \ (i = 1, \ldots, n), \text{ then } [[\tau]](a_1, \ldots, a_n)\ R\ [[\tau]](b_1, \ldots, b_n) .$$

In particular, then, items definable by means of closed lambda terms are relation-invariant. Conversely, Plotkin has shown that, at least up to the second-order level in the type hierarchy, all relation-invariant items must be lambda definable, in any structure. A concrete application of this notion will be found in Section 4.2 below.

Remark: Hereditary relations and their impact on lambda terms are reminiscent of the earlier notion of general inclusion. Nevertheless, there are some differences too, which preclude any general invariance result like the above with respect to Boolean inclusion. Special cases where the latter preservation does occur will be studied in the next Section. ♠

2.4 Polymorphism

Categorial derivation with its associated semantic interpretation as presented here may be viewed as a general mechanism for 'pooling' of denotational constraints contributed by the various components of an expression into the eventual semantic behaviour of the whole. Examples of this phenomenon have already occurred in the preceding Section. One suggestive focus for this phenomenon arises in the study of what is often called *polymorphism*, i.e., the ability of natural language expressions to modify their types in different environments, as required by the needs of interpretation. The above semantic account may also be viewed as a theory of polymorphism, explaining how items in one type can also occur in another one, with a systematic transfer of meaning:

$$\text{derivation of } a \Rightarrow b , \qquad \text{lambda term } \tau_b [\{x_a\}] .$$

In this perspective, a central question is one of 'dynamics':

> How are the earlier denotational constraints affected
> by type changing?.

Or, more generally,

> How does one constraint manifest itself across polymorphic type jumps?

Here are some answers.

Type change preserves *logicality*, in that permutation-invariant items in type a are transformed into permutation-invariant items in the derived type b . But, e.g., a *monotone* item in type a can lose its monotonicity in derived types.

Example: Loss of Monotonicity.
An individual A_e is trivially monotone - whereas, e.g., its derived form in type ((e, t), (((e, t), (e, t)), t)) is not:

$$\lambda x_{(e,\,t)} \cdot \lambda y_{((e,\,t),\,(e,\,t))} \cdot y(x)(A_e) \; . \qquad \spadesuit$$

In general, of course, there need not be exact reproduction of the original semantic property, but rather the emergence of some new variant of it. For instance, we have already seen in Section 2.2 how a property like *conservativity* for subject NPs (in type ((e, t), t)) also transfers, suitably understood, to object NPs in type ((e, (e, t), (e, t)) .

In addition to transferring or modifying already existing denotational constraints, polymorphism can also *create* new semantic properties of interest. For instance, the transition $e \Rightarrow ((e, t), t)$ transforms plain individuals A_e into *homomorphisms* in the NP type ((e, t), t) (Individual Lifting) :

$$\lambda x_{(e,\,t)} \cdot x(A_e) \; .$$

This 'semantic creation' behaviour of lambda terms may be investigated more systematically, using cues from logical Model Theory. What will happen in general is that such semantic behaviour may be read off from the *syntactic form* of the lambda terms. For instance, in the previous example, it is the fact of the variable x's occurring in the syntactic 'head position' of the lambda term which causes the homomorphic behaviour of the total function defined. Further examples will be introduced presently.

Another perspective on these issues arises by asking, not which *properties of* old denotations are preserved under polymorphism, but which *relations between* them continue to hold after the change.

Perhaps most obviously, there is the question as to which polymorphic changes reproduce the old items exactly, in that they establish a *one-to-one* map from some old type domain D_a into a new domain D_b. For instance, the above individual lifting is indeed one-to-one, when viewed as a map from items in the domain D_e (represented by the parameter 'A_e') to values in the domain ((e, t), t). Put differently, so that the analogy with the former kind of semantic transfer question becomes obvious:

> the lambda term $\lambda y_e . \lambda x_{(e, t)} . x(y)$ defines a one-to-one function in the domain of type (e, ((e, t), t))).

But e.g., the earlier reflexivization behaves differently:

> $\lambda x_{(e, (e, t))} . \lambda y_e . x(y)(y)$ does not define a one-to-one function in the domain of type ((e, (e, t)), (e, t)).

Here is an unsolved model-theoretic

Question: To give an effective syntactic characterization of the lambda terms defining one-to-one functions.

And here is another illustration.

Example: Equivalent Types.
A well-known categorial equivalence is that between basic predicates (e, t) and lifted intransitive verbs (((e, t), t), t):

$$(e, t) \Rightarrow (((e, t), t), t) \; : \; \lambda x_{((e, t), t)} . x(A_{(e, t)})$$

$$(((e, t), t), t) \Rightarrow (e, t) \; : \; \lambda x_e . B_{((e, t), t)}(\lambda y_{(e, t)} . y(x)) .$$

Nevertheless, the second transition is not one-to-one: and indeed, the two type domains are not isomorphic. ♠

Digression: One additional reason for interest in one-to-one definable maps is the possibility of *recreating* the domain hierarchy within itself at

some higher level. For instance, in setting up a viable semantics of *plurality*, one possible route is to switch from individuals in type e to *sets* of individuals (in type (e, t)) as the basic objects. This may be done without loss of earlier structure by mapping old individuals uniquely to their *singletons* in the new type:

$$\lambda x_e . \lambda y_e . y=x .$$

Likewise, the study of *polyadic quantification* sometimes requires viewing pairs or *sequences* of individuals as new basic objects (cf. de Mey 1988): which again involves an upward 'translation' of the universe. ♠

Remark: There are two equivalent ways of looking at the discussion so far. From a semantic point of view, there is little or no difference between studying definable type transitions $a \Rightarrow b$ or simply definable maps in the functional type (a,b) . Thus, an interest in various formalisms for defining polymorphic shifts is at the same time an interest in classifying denotations as to their logical complexity of definition. Accordingly, focussing on lambda terms $\tau[u]$ with parameters, or on the corresponding closed terms $\lambda x . \tau[x]$ is largely a matter of convenience. ♠

Another natural question of transfer concerns the earlier relation of general *inclusion*. If a polymorphic transition $a \Rightarrow b$ is to respect Boolean inference, then we must have the following general implication for the associated term τ (written with a harmless abuse of notation):

$$x_a \leq y_a \rightarrow \tau_b[x_a] \leq \tau_b[y_a] .$$

Note that this is the same as saying that τ , when viewed as denoting a function from the a-type domain to the b-type domain must be *monotone* in the earlier sense. Again, whether this property holds turns out to depend on the actual form of the lambda term involved.

Example: Preservation of Boolean Inclusion.
A typical example of a monotone function is given by the following term, describing a lifting of one-place sentence operators to one-place predicate operators:

$$\lambda x_{(e, t)} . \lambda y_e . u_{(t, (t, t))}(x(y)) .$$

A typical non-example is the earlier Individual Lifting

$$\lambda x_{(e, t)} .\ x(u_e) . \quad \spadesuit$$

Again, the logical question arises of characterizing just those lambda terms which induce monotone transitions in the required sense. Streamlining the treatment in van Benthem 1987a somewhat, we can say that a reasonable guess would be to allow only those terms in which the parameter u occurs only in the syntactic *head position*, as defined earlier on (which corresponds to the 'leading variable' of the matrix following the first adjacent prefix of lambdas). The latter certainly represent monotone mappings. But, there are some counter-examples to the converse, with typical cases such as the following (see the earlier reference):

(1) terms with arbitrary occurrences of individual parameters u_e

(2) iterations like $u_{(t, t)}(u_{(t, t)}(z_t))$.

So, the general characterization of monotone lambda terms is still open.

Digression: Reversing directions, it may also be asked which semantic behaviour would in fact be necessary and sufficient for having the parameter u occur in head position only. Here is a *conjecture,* based on a generalization of the earlier inclusion relation:

The following two assertions are equivalent for any lambda term $\tau[u]$:

(1) $\tau[u]$ has u occurring in its head position only ,

(2) $\tau[u]$ preserves 'local relations', in the following sense:
Let R be any relation on type domains satisfying the following condition on functional types:
for all $f, g \in D_{(a, b)}$, $f R g$ iff $f(x) R g(x)$ for all $x \in Da$.
Then always (with the earlier abuse of notation)
$u R v$ implies $\tau[u] R \tau[v]$.

As a small piece of evidence for the conjecture, it is easily seen how the above two kinds of counter-example disappear with preservation for arbitrary relations R . For instance, taking R to be the *inequality* relation on D_t , its lifted version will hold only between those functions in $D_{(t, t)}$ which are each other's complements. Thus, in the latter domain, the negation $\neg$ and the identity $\mathbb{id}$ are R-related. But, for any t-object x ,

it holds that $\neg(\neg(x)) = x = \mathbb{id}(\mathbb{id}(x))$; that is, the earlier monotone term does not denote R-related truth values. ♠

Remark: Model-Theoretic Preservation.
The above questions are related to the so-called 'preservation theorems' of logical Model Theory. But, there is one important difference. For instance, in the well-known preservation result for predicate-logical formulas that are semantically monotone with respect to some predicate, the resulting syntactic class is defined as follows. Modulo logical equivalence, the only relevant syntactic shapes are those formulas in which every occurrence of the relevant predicate is syntactically *positive*. Now, spelling out the definition of the latter notion, and taking an appropriate categorial view of predicate logic, the syntactic normal form class for monotone formulas turns out to include cases where the relevant predicate u occurs in embedded argument positions typically forbidden by the above type-theoretic description. The reason is this. Predicate logic has certain built-in logical constants, quantifiers and Boolean operators, which themselves stand for monotone items (whether 'upward' or 'downward'). And for instance, if u occurs monotonely in an argument term τ for a monotone function denoted by f, then its occurrence in the compound term $f(\tau)$ will still be monotone. A generalization of this predicate-logical situation to a lambda calculus having built-in monotone constants is immediate, and so is the analogous question of preservation. ♠

This concludes our discussion of the dynamics of polymorphic transfer of semantic behaviour.

Finally, it should be observed that there is more polymorphism in natural language than has appeared so far. What has been studied up till now is 'derivational' polymorphism, arising in the process of grammatical derivation. But, there is also evidence for 'variable' polymorphism, reflecting indeterminacies in initial type assignment. And other forms of type change may occur too, such as the various notions of 'collectivization' introduced in Section 4.1, needed to give a good account of plurality and related constructions. Naturally, such additional mechanisms raise their own questions of semantic transfer.

3. The Semantic Hierarchy

3.1 Fragments

Categorial Grammar comes with a landscape of weaker or stronger

calculi of implication as its combinatorial engines. And these calculi themselves form a hierarchy of ascending logical strength, which can be stratified according to various natural principles. But at present, we are interested in the *semantics* behind the enterprise. Now, there are various different kinds of semantics for conditional logics in the literature (cf. Gabbay & Guenthner, eds., 1984). But, as has become clear in the preceding Section, it is advantageous to look for the meanings of conditional proofs, rather than mere conditional assertions. And then, Type Theory is the general medium for expressing denotations associated with categorial derivations.

Now, this medium too admits of a significant layering. For instance, as with other logical formalisms, one could classify lambda terms according to specific syntactic patterns of their logical operators. A case in point are the so-called 'pure combinators' of the form

<prefix of lambdas; lambda-free application term>,

of which several have occurred in the preceding Section. And then, more complex lambda patterns could be classified. For present purposes, however, another principle of classification is more useful, namely by patterns of *variable binding*.

Again, there are several options here, but we shall follow a lead from an earlier discussion in Section 2.2. There, it was shown how the actual numbers of *occurrences* of variables in lambda terms reflect significant facts about semantic construction. In the full type-theoretic language, a lambda can bind any number of variables, but it is natural to look at more restricted cases too. For instance, one basic candidate is the fragment where each lambda binds *exactly one* variable, something which turns out to correspond to the basic categorial calculus of Lambek 1958, which has become a landmark in the area. (See van Benthem 1986 for the precise correspondence. Actually, the latter calculus has one additional restriction of 'non-empty premise sequents', which will be disregarded in this paper.) Thus, one basic hierarchy in categorial semantics is that of ascending fragments of Type Theory with ever increasing numbers of bindings allowed per lambda operator. (Here, the case with zero bindings could be identified with the pure application fragment.)

Many of the illustrations found in Section 2 already belonged to the single-bond Lambek fragment.

Example: Number of Bindings.
Here are two typical single-bond items:

$\lambda x_{(e,\,t)} .\ x(u_e)$ (Individual Lifting) ,

$\lambda x_{(e,\,t)}.\ \lambda y_{e}.\ u_{(t,\,(t,\,t))}(x(y))$ (Parametrizing Unary Sentence Operators) .

By contrast, the following typically require multiple binding:

$\lambda R_{(e,\,(e,\,t))}.\ \lambda x_{e}.\ R(x)(x)$ (Reflexivization)

$\lambda x_{(e,\,t)}.\ \lambda y_{(e,\,t)}.\ \lambda z_{e}.\ u_{(t,\,(t,\,t))}(x(z))(y(z))$ (Parametrizing Binary Sentence Operators) . ♠

Remark: Other principles of classification based on binding patterns are possible too - such as restricting the total *number* of available *bound variables* in advance. For instance, it can be shown that the expressive power of both the full Lambda Calculus and its Lambek fragment is essentially diminished by imposing a restriction to some fixed finite number of bound variables. Thus, the two principles of classification for lambda terms are really distinct. And, even further fine-structure of variable patterns might be investigated. Some examples will be mentioned below. ♠

Digression: The Role of Identity.
In this paper, 'Type Theory' has been equivalent, by and large, with 'Lambda Calculus'. Thus, we have not considered the richer formalism which would arise by adding *identity* to the above formalism. The latter enriched system is very powerful, and hence stands in even more need of fine-structuring into fragments. Nevertheless, it should be noted that the presence of identity affects the earlier binding hierarchy: as it may be used to simulate multiple variable binding. For instance, returning to predicate logic,

> In the presence of identity, lambdas need never bind more than *three* variable occurrences.

The reason is the trick displayed in the following example. Any formula of the form

$\phi(x, x, x, x)$

can be rewritten to

$\exists y_1 \exists y_2 \exists y_3 (\phi(x, y_1, y_2, y_3) \& x=y_1 \& y_1=y_2 \& y_2=y_3).$ ♠

As we shall see later on, fragments like the above may have much nicer semantic properties than the full type-theoretic language. In a general sense, of course, this phenomenon is already well-known from contemporary Logic. After all, the now 'standard' first-order predicate logic itself is a quite succesful fragment of full type theory, obtained by restricting attention to lower type levels. And within first-order logic again, fragments such as universal Horn clause formalisms have proved their surplus value in logic programming, precisely because of their special semantic characteristics.

Even so, one reasonable question is in which sense the above proposal can be said to form a genuinely *semantic* hierarchy.

Here, a distinction is to be drawn. What we have been mainly concerned with up till now is in fact *compositional* semantics, which studies merely the 'semantic glue' needed to merge component meanings into wholes. And by looking at weakest fragments needed to perform such functions, we are establishing the 'Price of Compositionality', so to speak. In addition, however, there is *lexical* semantics for denotations in specific categories. And these may come with their own special purpose classifications of semantic complexity: witness, for instance, the 'semantic automata' hierarchy of generalized quantifiers proposed in van Benthem 1986, 1987d. Even so, the border line between the two areas is fluid. For instance, type-theoretically definable items will form an important subclass, at least, inside many specific categories: and to them, the above classification applies directly. (Note that such items will be 'logical' in the sense of Section 2.2., which gives them an independent interest.)

Example: Classifying Logical Items.
Within the domain of each constructively provable type, we have type-theoretically definable objects, and hence, at least in principle, an ascending hierarchy of items definable by means of one-, two- or higher multiplicity of lambda binding. Thus, relation reducers in type ((e, (e, t), (e, (e, t)) include

single binding: $\lambda R_{(e, (e, t))}. \lambda x_e. \lambda y_e. R(x)(y)$ (Converse),

double binding: $\lambda R_{(e, (e, t))}. \lambda x_e. \lambda y_e. R(x)(x)$ ('Diagonal'),

etcetera .

♠

So, various layers of complexity of type-theoretic definition for semantic functions may be compared, either across different types, or even within a single one. As to the latter case, as was noted above, if a type is to have lambda-definable items at all, it must correspond to some constructive law of implication; and if it is to have, e.g., single-bond items, it will have to be provable already in the Lambek calculus. This still leaves many interesting cases for investigation, witness the function domain corresponding to the earlier categorial transition for complex transitive sentences

$$((e, t), t) \quad (e, (e, t)) \quad ((e, t), t) \quad \Rightarrow \quad t\,,$$

which was studied in Section 2.2. On the other hand, in the Lambek calculus, only the first reading would be available.

Finally, here are some further examples illustrating the workings of the binding hierarchy. One interesting question has to do, again, with the earlier distinction between compositional and lexical semantics. Natural language has 'invisible' mechanisms for performing certain compositional functions, so to speak, whereas other, more complex ones, have to be explicitly *lexicalized*. At least to a first approximation, it seems that single-bond Lambek transitions are freely available, whereas procedures for performing multiple binding have to be lexically expressed: witness the reflexivizer "self". (A possible counter-example are phenomena of deletion and gapping, where higher bindings may be at work, invisible in syntax.) Here is a technical illustration of the power of this division of labour.

Example: Quinean Predicate Logic.
In the 'predicate-functor' formalism for predicate logic proposed in Quine 1966, there are no variables, but only logical constants plus 'book-keeping' operators on predicates performing identifications and permutations. The latter formalism is provably as expressive as the usual Fregean one. Now, taking the appropriate categorial point of view on Boolean operators and logical quantifiers, predicate logic becomes a small fragment of Type Theory, which can be written in the following style:

usual notation: $\forall x(\exists y\, Rxy \;\&\; \neg Sxx)$

categorially: $\forall(\lambda x.\, \&(\exists\, (R(x)) \; (\neg(S(x)(x))))\,.$

Now, a restriction to single lambda binding here will produce a small fragment of predicate logic, studied , e.g., in van Benthem 1988c . (For instance, on a finite vocabulary, this fragment is logically finite.) The role of Reflexivization now becomes particularly vivid in the following

observation, which may be proved by an argument analogous to Quine's:

> Single-bond predicate logic with an additional reflexivizer of type ((e, (e, t)), (e, t)) is expressively equivalent with all of predicate logic.

For the full type-theoretical language, no similar result holds. It will be necessary to add reflexivizers in all types ((a, (a, b), (a, b)) . Provably, no finite number of these will do. [The reason is this. Such a finite set of reflexivizers has only a finite corresponding set of 'type counts', in the sense of van Benthem (1986, chapter 7). But, it will have to derive all the above contraction types in the Lambek calculus: which preserves type counts. This contradicts the fact that the above family of contractions has an infinite associated set of type counts.] ♠

There is still a lot more room for speculation here on the border line between logic and linguistics. For instance, natural language does not have arbitrary reflexivizers, even where these would be a priori possible. For instance, one could imagine a reflexivizer self on binary relations over unary predicates, such as determiners: so that "Every self dog" would stand for "every dog is a dog". But, such particles do not seem to occur. (Of course, one can think of ad-hoc reasons why having this particular feature would not be all that useful in natural language.)

Another point of fact is that not all of the single-bond mechanism appears to be freely available after all. Notably, various forms of *permutation*, a feature built into our semantic Lambek calculus, are in fact lexicalized in the passive constructions of natural language. One possible reason here is that performing permutations in type-theoretical terms requires stacking of lambdas: and, judging from the examples in Section 2, there may be a natural tendency to avoid these. If this is taken seriously, we might have yet another source of semantic fine-structure, namely various restrictions on the length or depth of lambda nesting.

3.2 Expressive Power

Next, there are various more theoretical issues to be considered in the new finer-grained perspective. Notably, what happens to the semantic theory developed in Section 2 as one varies the semantic formalism along the different fragments introduced above? Here are some samples.

First, the availability of *readings* for expressions will be obviously affected by the expressive power of the semantics.

Example: Readings Revisited.
Returning to the main example of Section 2.2, here are some new outcomes (with 'I' standing for the full Type Theory corresponding to intuitionistic derivation, 'L' for the Lambek single-bond fragment, and 'A' for the pure application fragment corresponding to the original categorial grammar of Ajdukiewicz) :

(1) the transition e (e, t) $\Rightarrow$ t has exactly one reading in A , L and I .
(2) the transition e $\Rightarrow$ ((e, e), e) has no reading in A , exactly one in L , and infinitely many in I .
(3) the transition (t, t) t (t, t) $\Rightarrow$ t has exactly two readings in A , L and I .
(4) the transitions from 'n copies of e to e' have no readings in A or L , and exactly n in I ; except for the cases n=1 (just one reading everywhere) and n=0 (no readings anywhere).
(5) the transition for complex transitive sentences has no readings in A , exactly four in L , and infinitely many in I .

♠

In particular then, the full lambda calculus allows an *infinity* of readings in some cases, whereas this did not happen in the Lambek fragment. There is a general result behind this:

Proposition: The number of single-bond lambda terms $\tau_b [\{x_a | a \in A\}]$ for any L-transition $A \Rightarrow b$ is *finite* up to logical equivalence.

The proof of the latter result is constructive (cf. van Benthem 1986): we can find representatives of each equivalence class. Thus, as a corollary, we can also determine such special cases as the earlier 'unambiguous' transitions having a *unique reading*.

Nevertheless, the number of Lambek readings may still be subject to combinatorial explosion. For instance, in the following sequence of stacked prepositional phrases

"to NP_1 [from NP_2] ..... with NP_k"
(e, ((e, t), (e, t))) ((e, t), t) ..... (e, ((e, t), (e, t))) ((e, t), t)
$\Rightarrow$ ((e, t), (e, t)) ,

we can extract quantifiers in any order, yielding k! genuine scope ambiguities. And even weaker calculi, designed expressly to curb this

tendency, such as that of Hendriks 1988b, though more parsimonious, still seem to involve exponential growth. It would be of interest to have more precise numerical information in this area, which could then be measured against possible general computational intuitions concerning the semantic complexity of natural language.

Another area of interest are the earlier questions of *polymorphic transfer* and their related preservation results. For instance, the natural syntactic conjecture for *monotonicity* of lambda terms, amounting to having the relevant parameter only in head position, which failed for the full type-theoretic language, does in fact hold for the Lambek fragment (cf. van Benthem 1987a). Similar improvements, in the form of model-theoretic preservation theorems, are probably possible for such notions as 'one-to-one mapping', and other cases of interest in Section 2.

Here again, we want to conclude with the general question as to the genuine *semantic* content of our framework. Is not relating denotational constraints to specific lambda forms rather doing a form of logical syntax?

There are various points to be noted here. Generally speaking, logical Model Theory is about the interplay between certain syntactic forms of definition and certain semantic forms of behaviour. Thus, the preceding questions would be typically model-theoretic in this sense. Still, even on this point of view, one would like to see, conversely, some matching semantic forms of behaviour for the various fragments of full type theory introduced above. And in principle, this ought to be possible. But, existing model-theoretic notions seem designed for some kinds of 'fragmentation' rather than others. In particular, they fit in better with divisions according to occurrences of logical operators, and not so much with those according to binding patterns.

Example: Characterizing Pure Combinators.
A typical fragment of the former kind would be the earlier pure combinators, consisting of some lambda prefix followed by a lambda-free matrix. And indeed, it is an interesting (open) question to characterize these forms, amidst all type-theoretical terms, by some notion of invariance across suitable 'substructures'. (In general, formulating such results will probably force one to leave the standard function hierarchies of Section 2.1, and move to some 'generalized model' version thereof: as is quite common in the literature.) ♠

But, a more radical reaction is possible too. Perhaps, the present perspective of meanings for derivations has really changed the traditional

notion of 'semantics': by making *complexity* of truth conditions, or their associated verification procedures, an integral part of the enterprise. If this is so, then there ought to be significant differences in complexity between, say, the L fragment and the full I formalism. And in one sense, this is what the earlier observations indicate. But, how to to substantiate such claims by means of some formal interpretative procedure?. Here, some generalized version of the 'linear machine' of Lafont 1988 may be useful. Another possibility derives from the model-theoretic notion of *Ehrenfeucht-Fraisse games* , suitably augmented with further game structure, such as the *pebbling* used in Immerman 1982 for characterizing variable-restricted fragments of predicate logic. We shall illustrate this for the above single-bond predicate logic.

Example: Model-Comparison Games for Single Binding.
In an Ehrenfeucht-Fraisse game, two players are comparing two models, with player I trying to establish their difference, player II their similarity. The game proceeds in rounds, started by I who picks an object in one of the models, and followed by II who picks a (matching) object in the opposite model. After some pre-arranged number of rounds, a comparison is made between the two finite submodels obtained, by collecting the objects chosen on each side. If these are isomorphic, then player II has won, otherwise, the win is for player I . The basic result here is the following:

> Two models verify the same predicate-logical sentences up to quantifier depth n if and only if player II has a winning strategy in their Ehrenfeucht-Fraisse game over n rounds.

In fact, the final comparison itself may be pictured as another game: player I is allowed to choose any fact involving the objects selected, whose truth value is then compared in both models. All comparisons must match, if player II is to have a guaranteed win here.

Now, Immerman's characterization of restricted bound variable fragments of predicate logic involves changing the course of the main game: by allowing selection of objects only through some fixed amount of *pebbles* which can be put on them. When these run out, earlier selections will have to be undone by removing a pebble.

By contrast, in order to characterize the single-bond fragment, a restriction is needed, not on the main game, but on the comparison to be performed during the 'end game':

1 the fact selected can only involve *distinct occurrences* of objects selected in the course of the main game (represented, say, by their 'stage numbers')

2 this fact has to be selected by I *beforehand*.

Here, condition 1 is inspired by attention to the actual mechanics of verification. Because of it, typically, the players will not be able to see the difference between a reflexive model, satisfying the double-bond property $\forall x\, Rxx$, and one which is not reflexive. Condition 2 , on the other hand, serves to break any Boolean connection between atoms in the matrix, thus removing another source of multiple binding (exemplified in formulas like $\forall x(Px \vee Qx)$). Call this new convention the 'Linear Token Game'. Then, the following can be proved by an obvious induction:

> Two models verify the same single-bond formulas up to quantifier depth n if and only if player II has a winning strategy in the Linear Token Game on these models over n rounds.

Evidently, there are further possibilities for playing around with Ehrenfeucht-Fraisse games. For instance, the joint effect of condition 1 with fact selection postponed until the end yields another interesting fragment of predicate logic. ♠

Of course, this analysis is to be extended to Type Theory in general. Possibly, this can be done in terms of players picking corresponding objects on either side, so that all possible equalities between application compounds of these will match on both sides in the final comparison.

But, other kinds of semantic games might be worth exploring in this context too: such as *Hintikka verification games* on single structures.

Digression: Complexity of Proofs.
'Complexity' is a phenomenon which can be conceptualized in many different ways. And therefore, outcomes may not always be unambiguous. Thus, the Lambek system will not do better than the full intuitionistic one on *every* count of proof complexity. For instance, one initial conjecture might be that, in each type which is L-(and hence I-)derivable, there will be smallest L-derivations. But, this need not be so. For instance, here is the L-derivation again for the earlier example $e \Rightarrow ((e, e), e)$, together with a shorter I-derivation, employing a structural rule ('Empty Conditionalization') which is not available in L :

$$\frac{e \qquad (e, e)}{e}\,\text{MP} \qquad\qquad \frac{e}{((e, e), e)}\,\text{COND}$$

$$\frac{e}{((e, e), e)}\,\text{COND}$$

And similar examples may be found with cases of L-derivable Contraction, such as $((t, t), ((t, t), c) \Rightarrow ((t, t), c)$. ♠

Finally, one other semantic road might leave the idea of interpreting *derivations* for a while, concentrating on mere derivable *sequents*. This, after all, is what happens in ordinary semantics of logical calculi, where one abstracts from the derivational motivation for valid inferences. Thus, we arrive at the question what independent viewpoints are possible on the implicational calculi in the categorial landscape. As has been observed before, these can be modelled in various ways: algebraically, in possible worlds style, etcetera. This issue will return in Appendix 5.2, where a connection is surveyed with current 'Linear Logic', an enterprise whose motivation is in fact quite close to the present one.

4. Variations

The above investigation has been concerned with the fine-structure of standard type-theoretic semantics (as originated by Montague), a theory which is undergoing considerable modification at present. And in fact, of that old standard theory, only the so-called 'extensional fragment' has been considered so far. In this Section, a number of possible extensions or modifications of the earlier approach will be considered, in order to show that the just-mentioned restrictions can be liberalized. These more liberal set-ups may be grouped loosely under the headings of 're-interpreting old types' and 'introducing new types'.

4.1 Re-Interpreting Individuals

It has often been pointed out that the bare structure of the individual domain D_e cannot suffice for many descriptive purposes. Thus, a viable account of the behaviour of, e.g., *mass terms* requires a domain of individuals structured, at least, by some form of non-Boolean inclusion. Likewise, in the treatment of collective quantification, it seems imperative to take *groups* into consideration, in addition to single individuals.

One somewhat conservative move, which already provides a great deal of useful additional structure, is to replace the individual type e by the type (e, t) of sets of individuals in a number of categorizations - as has been advocated by various authors since the early eighties. Notably, *intransitive verbs* will now receive type ((e, t), t) [formerly, that of NPs], rather than (e, t) - whereas one might retain the old type (e, t) for *common nouns*. (But, e.g., van Eyck 1985 has an e-to-(e, t) replacement throughout.)

As we have seen in Section 2.3, such one-to-one self-embeddings of the universe preserve a lot of the original structure. Even so, re-categorizations such as these ask for adjustment of previous denotational constraints for lexical items. But apart from that, the earlier general derivational apparatus will work in the same fashion. Here is an illustration.

Example: Adjusting Generalized Quantifiers.
Determiners may now be taken to live in the following type

$$((e, t), (((e, t), t), t)) ,$$

relating sets with families of sets. Truth conditions which have been proposed then exhibit such forms as

ALL AB : (1) $A \subseteq \cup \{ X \cap A \mid X \in B \}$ (van Benthem 1986)
(2) $A \subseteq \cup \{ X \subseteq A \mid X \in B \}$ (Hendriks 1988a)

In fact, these exemplify 'lifting strategies' for the old determiners, such as

(1) D_{old} goes to $\lambda A_{(e, t)} . \lambda B_{((e, t), t)} . D_{old}(A, \cup B \,\text{`}A)$

(2) D_{old} goes to $\lambda A_{(e, t)} . \lambda B_{((e, t), t)} . D_{old}(A, \cup(B \cap pow(A)))$.

Earlier denotational constraints also return. In particular, *Conservativity* has even been built into the above truth conditions.

What happens in general, however, is that the additional structure on the new 'individual' domain becomes important. In the present case, this is mainly *set inclusion*. For instance, this time, we should expect *Logicality* of determiners to mean, not invariance for arbitrary permutations of the new individuals (possibly disrupting their relative location), but only with respect to *inclusion automorphisms* of $D_{(e, t)}$,

satisfying the following equivalence:

$$\pi(X) \subseteq \pi(Y) \quad \text{iff} \quad X \subseteq Y .$$

Still, the latter may be represented again as being induced by *arbitrary* permutations π^* on the old individuals, obtainable from them as follows:

$$\pi^*(x) \;=\; \cup\, \pi(\{x\}) .$$

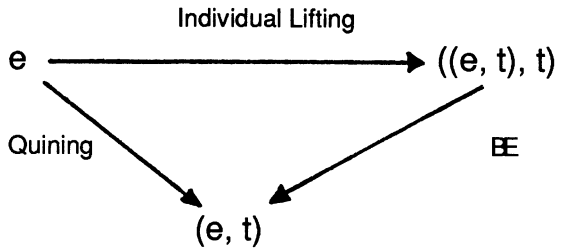
And thus, the earlier notion of logicality has really survived in the same mathematical form.

Nevertheless, the additional structure can be more interesting too. For instance, *Monotonicity* will now come in several variants, because there are other natural relations between families of sets than mere set inclusion. Thus, the above two clauses (1), (2) both satisfy the 'straightforward' variant of monotonicity in the B argument: if true, they remain true for larger families of sets B . But, we can also define

$$B_1 \leq B_2 \quad \text{iff} \quad \forall X \in B_1\, \exists Y \in B_2\; X \subseteq Y .$$

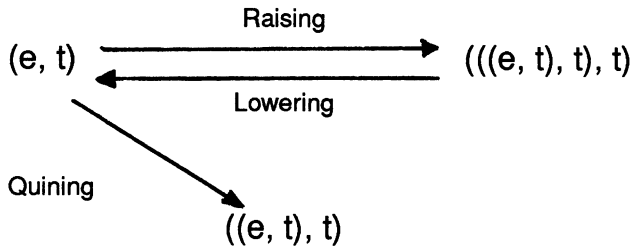
Reading (1) is monotone with respect to this relation too, whereas reading (2) is not. Thus, a new kind of semantic variety is encountered. ♠

Another useful perspective on the polymorphism occurring here is given in the following 'Partee Triangle' of related basic types, with their various semantic interconnections:



Here, 'Quining' is the earlier operation of forming singletons. See Partee 1987, or van Benthem (1986, chapter 3), on the applications and the theory of this schema. For instance, note that the various polymorphic operations in the Triangle commute.

Example: 'Cosy Corners'.
Such linguistically interesting inter-connected corners of the type-theoretical universe occur more often; witness, e.g., another triangle, proposed by Henk Verkuyl in the study of collectives:



Here, Raising is the obvious analogue for predicates of Individual Lifting. But this time, unlike in the previous case, there is also a lambda-definable return road (as was observed already in Section 2.3), being the Lowering map

$$\lambda x_e .\ u_{(((e,\,t),\,t)}(\,\lambda z_{(e,\,t)} .\ z(x))\ . \qquad \spadesuit$$

4.2 Re-Interpreting Truth Values

Another broad source of semantic variation is to be found in various enrichments of the truth value domain beyond the two values 0 and 1 . For instance, recent *partial model theories* employ three or four truth values (cf. Muskens 1988), whereas *fuzzy logic* employs even a whole continuum (cf. Zadeh 1988). Again, such a move does not affect the earlier categorial apparatus as such - although it may engender more sophisticated versions of earlier semantic definitions. (See J-E Fenstad et al. 1987, or van Benthem 1988a, on some issues in partial Type Theory, and Ketting 1988 on the theory of fuzzy generalized quantifiers.)

But, even the *intensionality* already present in Montague's framework may be viewed as arising through a re-interpretation of the old type t . This now becomes a domain of *propositions*: which can afterwards be represented as sets of possible worlds, in a new type (s, t) ; where

D_t is again {0,1} in the old sense,
D_s is a new primitive domain of 'possible worlds', or 'indices' .

This time, there is a new issue of categorial combination, namely as to how freely the new s type may participate in semantic derivation. In fact, it seems that no Lambek single-bond restrictions apply to bindings of the form λx_s (cf. van Benthem 1988d). For instance, in reading an intensionalized version of the sentence

"all	elephants	danced"		
((e, t), ((e, t), t))	(s, (e, t))	(s, (e, t))	⇒	(s, t) ,

one might want to end up with the following multiple binding

$$\lambda x_s .\ \mathrm{ALL}(\mathrm{ELEPHANT}(x))(\mathrm{PAST}(\mathrm{DANCE}(x)))\ .$$

But again, perhaps the more interesting semantic questions have to do with the intricacies of the new semantic situation as such. Here is one illustration - which can be dealt with eventually by means of already available type-theoretic techniques.

Example: What is Extensionality?.
One surprising feature of intensional semantics, not often brought up, is that we apparently lack a clear-cut *general* notion of *extensionality* in an intensional Montagovian universe. There are some paradigmatic *examples*, of course. For instance, a predicate is sometimes called 'extensional' if its interpretation does not change over possible worlds; i.e.,

P in type $(s, (e, t))$ is of the form $\lambda x_s .\ P^*_{(e, t)}$,
for some extensional predicate P^* in type (e, t) .

Or, an adjective A in type $((s, (e, t)), (s, (e, t)))$ is called 'extensional' if the computation of its value for some intensional predicate P is 'local':

$A(P) = \lambda x_s .\ A^*(P(x))$,
for some extensional adjective A^* in type $((e, t), (e, t))$.

But, there seems to be no obvious road from these examples towards answering the question when an object in some *arbitrary* intensional type $a = a(e, t, s)$ is 'extensional'.

Nevertheless, an answer may be proposed in terms of the categorial notions developed in Section 2 (see especially Section 2.3). It can take either a 'linguistic' or a 'structural' form.

Linguistically, call a term *l-extensional* if there exists a *definition* for it, in the form of a lambda term of the e, t, s type theory whose parameters carry only pure e, t types. The intensional types a in which such items occur will be those which are derivable in the intuitionistic conditional logic from a sequence of pure e, t types.

Structurally, call an object in some domain *s-extensional* if it is *invariant* for *individual relations* on D_s (leaving D_e, D_t undisturbed). (See

Section 2.2 for this notion.) Thus, given fixed domains of individuals and truth values, it can survive drastic changes in the possible worlds structure. This seems to make sense in general semantic terms, while producing the right results in the above special cases. (For a technical proof, see van Benthem 1988f.)

In general, l-extensionality implies s-extensionality, but not conversely (cf. Plotkin 1980). So, there is still an interesting option. ♠

Variations such as the above will be necessary, in particular, when the type-theoretic framework is used for dealing with *programming languages*, where the new s-domain represents the *states* of a computing device. (This the dynamic logic approach surveyed in Harel 1984. Cf. also van Benthem 1988*.)

4.3 New Types

In the last analysis, one also needs entirely *new* kinds of type for semantics. Examples are provided by the many recent *event*-based frameworks, where individuals and events are treated on a par, occupying separate (though formally similar) primitive domains D_e and D_E. Even so, this need not invalidate the earlier categorial approach. For instance, one interesting line of investigation arises with the new basic sentential pattern (note the analogy with the earlier scheme for plurality)

Det	CN	VP	
((e, t), ((E, t), t)	(e, t)	(E, t)	⇒ t.

What will be the proper 'temporalized' account of generalized quantification, and its denotational constraints?

Here is another kind of new feature which has been proposed, this time not concerning the introduction of new types, but of new modes of categorial combination. (In general, this is not at all against the grain of a type-theoretical approach: witness Lambek & Scott 1986.)

Digression: Coordination versus Subordination.
In connection with event-based frameworks, it has sometimes been suggested that another basic combinatory mechanism should be envisaged, in addition to *function application* - namely some form of *coordinating conjunction*. For instance, many relative clauses and adverbial or adjectival phrases might be treated in the latter manner, rather than in the earlier subordinating applicative mode.

This is indeed possible. But, there is also an alternative: which is to employ the old functional apparatus provided with suitable *denotational constraints*. For instance, an ('extensional') conjunctive adjective is a function A in type ((e, t), (e, t)) which satisfies the following conditions:

it is introspective: $A(X) \subseteq X$,
it is continuous: $A(\cup_{i \in I} X_i) = \cup_{i \in I} A(X_i)$.

And, all such adjectives can be *represented* in the following conjunctive form (cf. van Benthem 1986) :

$A(X) = X \cap A^*$
for some predicate A* of individuals. ♠

Another, more fundamental change would be to allow *variable types* in our set up - as has become widespread in computational linguistics, with *unification* serving as a major mechanism for building up and passing on information.

Apart from computational convenience, there may also be intrinsic semantic motivations for taking this line. For instance, in many cases, the above e, t, s types are too rigid: even granting the phenomena of derivational polymorphism discussed in Section 2. An illustration may be found in the earlier area of collective expressions and *plurality*. We can start a sentence "all bandits" thinking indeterminately of individual bandits or groups of these, letting a choice occur only after having encountered a subsequent predicate: distributive ("snored"), or collective ("quarrelled"). Being able to assign a variable-based type

((x, t), t)

to such a head NP would leave the semantic choice open until the appropriate moment. (But note that this is an argument about semantic *procedure* rather than semantic *content*.)

Another example arises with *intensionality*. For instance, in the earlier treatment of a sentence like "All elephants danced" , we let the determiner have its standard type ((e, t), ((e, t), t)) : requiring it to capture extensionalized 'snapshots' of the two (s, (e, t)) predicates involved. But in certain contexts, this determiner might have a stronger force, expressing a regularity across possible worlds: that is, in type

((s, (e, t)), ((s, (e, t)), t)) .

Again, having an initial assignment employing an appropriate type variable would give us this intuitive latitude right from the start. (Note how this idea is also relevant to the practice of logical *inference*: where it may be only the context of a proposed inference *itself* which gives us the required concretization.)

If this move is to be more than just a convenient computational trick, however, we shall have to devise an independent semantics for it. Here, suggestions might be taken from the *category-theoretic* approach of Lambek and Scott 1986, or the parametrized structures of Situation Semantics, which build indeterminates right into their semantic objects. This would certainly require an overhaul of the earlier standard Type Theory. (See van Benthem 1989 for more detailed discussion of the intrinsic case for variable polymorphism.)

5. Appendices

5.1 Generating Readings

As was already observed in Section 2.2, techniques for obtaining enumerations of readings have an independent interest. Here we present one general method, based on formal grammars and automata, which was developed in van Benthem 1987c in order to classify possible readings for complex transitive sentences.

This time, however, we take another example, concerning the category of *determiners*. Moreover, we shift the question somewhat: since determining possible readings is merely one way of looking at the issues involved. (Compare the digression 'for and against occurrences' at the end of Section 2.2.) Thus,

> When is a semantic determiner, in type ((e, t), ((e, t), t)),
> lambda-definable from items in lower types
> [i.e., e, t, (e, t), ((e, t), t)] ?

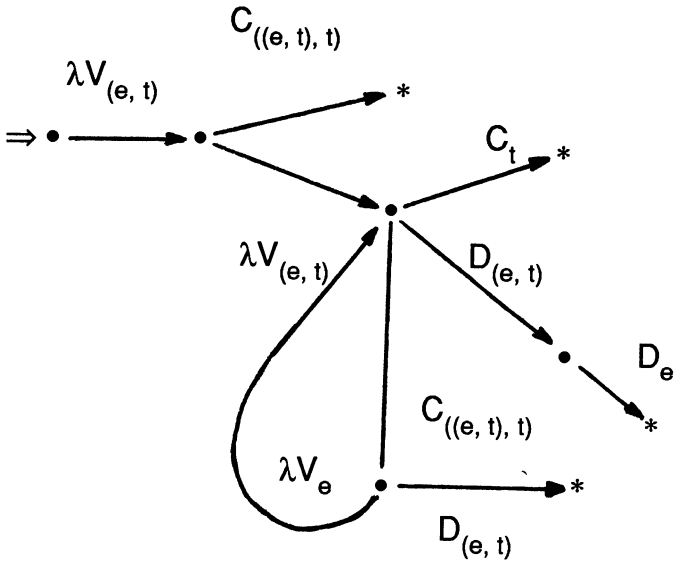
We can describe all possible lambda normal forms for determiners with parameters from the types indicated using a kind of *context-free grammar*, having symbols (at most) X_a, X_a^*, C_a, V_a for each of the relevant types a. Here, V_a stands for a variable of type a, C_a for a constant (parameter), X_a for any term of type a, X_a^* for such a term which does not start with a lambda. The point of this division will become clear from the rewrite rules for terms in normal form to be presented now, for all types a:

$$X_a \Rightarrow X_a^* \quad ,$$
$$X_a^* \Rightarrow C_a \quad ,$$
$$X_a^* \Rightarrow V_a \quad .$$

Next, rules for application or lambda abstraction depend on the actual types being present. [Recall the fact that lambda *normal forms* contain no more redexes of the form $(\lambda x.\ \alpha)\ (\beta)$, while the types of all variables occurring in them must be subtypes of either the resulting type or one of the parameter types.]

$$X_{((e,\,t),\,((e,\,t),\,t))} \Rightarrow \lambda V_{(e,\,t)}.\ X_{((e,\,t),\,t)}$$
$$X_{((e,\,t),\,t)} \Rightarrow \lambda V_{(e,\,t)}.\ X_t$$
$$X_t \Rightarrow X_{(e,\,t)}^*\,(X_e)$$
$$X_t \Rightarrow X_{((e,\,t),\,t)}^*\,(X_{(e,\,t)})$$
$$X_{(e,\,t)} \Rightarrow \lambda V_e.\ X_t \quad .$$

The description of possible readings is much facilitated here, because we can make this grammar *regular*. This may be visualized in the following *finite state machine* for producing readings (where 'D_a' stands for C_a or V_a, where applicable):



This scheme produces determiner denotations of forms such as the

following:

1. $\lambda x_{(e, t)} \cdot c_{((e, t), t)}$
2. $\lambda x_{(e, t)} \cdot \lambda y_{(e, t)} \cdot c_{((e, t), t)}(x)$
3. $\lambda x_{(e, t)} \cdot \lambda y_{(e, t)} \cdot x(c_e)$
4. $\lambda x_{(e, t)} \cdot \lambda y_{(e, t)} \cdot c_{((e, t), t)}(\lambda z_e \cdot c'_{((e, t), t)}(\lambda u_e \cdot x(z)))$

Here, the latter kind is 'iterative', producing infinitely many forms by repeating the $c'_{((e, t), t)} (\lambda u_e$ subroutine. Thus, globally, there are *infinitely* many distinct possibilities for defining determiners.

Nevertheless, this global infinity is still 'locally finite': a phenomenon already observed in Section 2.2. For, all these forms are equivalent, in any model, to only a *finite* number of cases.

The reason is as follows. Any scheme of definition will start with an initial $\lambda x_{(e, t)}$, and then some further term τ of type $((e, t), t)$. Now, if the latter contains no free occurrences of the variable $x_{(e, t)}$, then it defines some fixed object: which also has one parameter $c_{((e, t), t)}$ denoting it. Hence, we are in the above case 1. Next, if the variable $x_{(e, t)}$ does occur in τ, then, analyzing the latter term one step further, we can rewrite the whole scheme of definition to

$$\lambda x_{(e, t)} \cdot \lambda y_{(e, t)} \cdot [(\lambda z_{(e, t)} \cdot \tau[z/x]) (x)] ,$$

where the subterm $\lambda z_{(e, t)} \cdot \tau[z/x]$ does not contain any free occurrence of the variable $y_{(e, t)}$. (To see this, check the 'exit routes' in the above diagram.) Now, this subterm again denotes some fixed object in the $((e, t), t)$ type domain: and hence, we arrive at the above form 2.

Therefore, the general result becomes this:

Terms of the first two kinds listed above always suffice. ∎

This result says that determiners admit of no non-trivial reductions to lower types: they are a genuinely new semantic category.

Evidently, this is just one of a number of questions about

reducibility in type domains which may be investigated. For instance, can we prove general 'hierarchy results' on definability?

Remark: Readings Revisited.
Actually, a proof of the local finiteness result for polyadic quantifier readings, referred to in Section 2.2, requires a more careful analysis of the potential infinity of lambda forms, relying on Boolean reductions to arrive at a finite list. ♠

Whether for this purpose or for others, the 'grammatical' approach to description of possible type-theoretic readings may be of interest by itself. For instance, here is one

Conjecture: A derivable transition has *finitely* many readings
if and only if its associated context-free grammar is *acyclic*.

5.2 Linear Logic

The Type Theory employed in this paper for categorial purposes has only few connectives, namely 'implication' , and 'conjunction' . . From a logical point of view, this makes it a rather poor system. Is there no use for other connectives, and hence a richer logic?

In fact, there have been various proposals in the literature for making linguistic use of additional type structure. For instance, *conjunction* types would enable us to say that an expression belongs to two categories at the same time. Moreover, *disjunction* types would encode partial information about an expression's belonging to one of a number of possible categories. (Compare 'disjunctive feature structures' in lexically oriented grammars.) Finally, Moortgat 1988 has various 'infixation operators' for categorial encoding of linguistic expressions with gaps.

A more systematic linguistic perspective on these matters arises in the theory of *formal languages*. In fact, we can also interpret a calculus like Lambek's as referring to the behaviour of concatenation of expressions in different 'languages' (being the sets of expressions of different categories; cf. Buszkowski 1982):

$$x \in L_{a.b} \quad \text{iff} \quad \exists x_1 \in L_a \, \exists x_2 \in L_b \; x = x_1 {}^\frown x_2 \qquad (L_{a.b} = L_a . L_b)$$
$$x \in L_{a\backslash b} \quad \text{iff} \quad \forall z \in L_a \; z {}^\frown x \in L_b$$
$$x \in L_{b/a} \quad \text{iff} \quad \forall z \in L_a \; x {}^\frown z \in L_b .$$

I.e., \ and / are left- and right-inverses, respectively, of the concatenation product . . Thus, the Lambek calculus encodes part of the behaviour of language families under useful operations. But then, there are many further candidates which qualify; such as, in particular, all *Boolean* functions of intersection, union and complement. But also, noting that all these examples are natural operations staying inside the class of regular languages, one might add the Kleene iteration * . And in fact, the latter has an intrinsic linguistic motivation too. For, when dealing categorially with *texts*, arbitrary finite iterations of sentences and operations over these will have to be considered in any case.

Digression: Closure Operations on Regular Languages.
Products, Unions and Iterations are the canonical operations on regular languages in the Kleene normal form theorem with respect to finite automata. But, arbitrary Booleans make equally good sense. And in fact, the two Lambek implications are closure operations on regular languages too. Here is a sketch of the argument for the leftward slash a\b :

Suppose that M_b is a finite state recognizer for L_b . Now, localize all states in this machine which can be reached, starting from its initial state, by processing some string belonging to L_a . Then, an arbitrary string will belong to $L_{a\backslash b}$ if and only if it will drive M_b to a recognizing state from each of those 'L_a-reachable' states. And the latter condition can be checked by a finite state automaton constructed out of M_b in an obvious way. ♠

Just as with the original implications and products, a proof calculus may be set up for the new connectives. For instance, Boolean conjunction behaves as follows:

$$\frac{A \Rightarrow b \quad A \Rightarrow c}{A \Rightarrow b \cap c} \qquad \frac{A, b \Rightarrow d}{A, b \cap c \Rightarrow d} \qquad \frac{A, b \Rightarrow d}{A, c \cap b \Rightarrow d}$$

There is a slight analogy here with concatenative conjunction; but, there are also obvious differences. E.g., typically, Boolean conjunction is 'monotonic' on the left, concatenative conjunction is not.

For the Kleene star, however, no standard introduction and elimination rules lie at hand. But, it will still satisfy such principles as

$$A^*, A^* \Rightarrow A^* , \qquad A \Rightarrow A^* .$$

(The situation may be compared with the axiomatization problem for regular programs in propositional Dynamic Logic: cf. Harel 1984.)

Very important in setting up such calculi is the use of a weak categorial base, like the Lambek calculus. The latter may be described as a system of implicational logic agreeing with ordinary systems in its 'logical rules', while leaving out the so-called *structural rules* for handling derivable sequents (cf. van Benthem 1986 on this perspective). The latter feature reflects the fact, discussed already in Section 2.2 above, that, in a categorial grammar for analyzing linguistic expressions, one deals with *occurrences* of premises rather than equivalence classes over these, as happens in standard logic. This weakness allows for distinctions such as the one between the above two conjunctions: which would collapse into one given the full structural rules of standard logic.

Remark: A Hierarchy of Structural Rules.
Much hierarchical structure in the categorial landscape arises from a ladder of possibilities starting from some calculus with no structural rules at all, and then adding various structural extras, such as Permutation, Contraction or Thinning of premise sequents: all the way up to the full calculi of ordinary logic. This view fits in with the semantic hierarchy of Section 3. For instance, the rule of Contraction, being

$$\frac{A, b, b \Rightarrow c}{A, b \Rightarrow c} \quad ,$$

will bring in lambda terms for derivations in which multiple variable binding is allowed after all. ♠

Now, the same perspective may also be arrived at from a purely logical angle, having to do with the proper analysis of reasoning. Recent work on so-called 'linear logic' (cf. Girard 1987, Lafont 1988 or Avron 1988) has started from a criticism of standard logical systems, whose structural rules treat the premises of an inference as a mere set, destroying any available information about their actual organization. More sensitive systems of logic, respecting this structuring of premises, will have to drop the standard 'structural rules' (who, in a sense, are rather *de-structuring* operations). Thus, Girard has come to advocate a basic 'occurrence logic', whilst also stressing its potential for making finer distinctions among logical constants. That is, in linear logic, there are two conjunctions (as above) and, at least in some systems, also two

disjunctions. Moreover, one very fruitful idea has been to absorb some of the usual structural rules into *new logical constants*. For instance, Contraction is not allowed in general; but it is admissible for so-called exclamation forms $!\phi$ (standing for 'arbitrary occurrences of ϕ'), whose logic is more classical. This new operator resembles the above Kleene star *, which also allowed derivable contraction. (It may even be exactly the same notion.) There are many interesting technical results about the various systems of linear logic, which we cannot go into here. As a final point of motivation, again, it may be noted that there are also complexity considerations behind the new enterprise which would seem to be quite congenial to those underlying the Semantic Hierarchy of Section 3 above.

More generally, this convergence between grammatical and logical concerns, which can be observed to-day, may be understood as follows. Categorial grammarians are looking for systems describing syntactic structures, whilst also insisting on some principled semantic motivation. This leads to calculi taking a somewhat liberal attitude towards every last syntactic detail. Thus, from the Ajdukiewicz system upward, one ascends toward stronger logical calculi of types. Linear logicians, on the other hand, are starting to take the actual structure of premises more seriously. This takes them on a downward road, from full standard systems to weaker ones, more sensitive to syntactic detail, but hence also: to logical distinctions between logical constants. And, the two movements turn out to meet on common ground. (It now remains, of course, to turn a casual encounter into real affection, and a life-time of collaboration.)

Finally, once more, there remains an important issue concerning these calculi of linear logic, namely:

What is their genuinely *semantic motivation*?

To be sure, it has turned out possible to devise various algebraic modellings, both in categorial grammar and in linear logic. And also, one can expand the earlier lambda calculus so as to have analogues for the new logical constants. But, in a sense, these are just various direct *encodings* of what remain essentially *proof-theoretically* motivated systems. At least for those, then, who do not believe that Proof Theory is the genuine Theory of Meaning, a task remains. What would be an independently motivated, enlightening semantics for Lambek or Girard type calculi?

We conclude by presenting one kind of modelling which may at least point in the right direction. Broadly speaking, inference in standard systems of logic describes an inclusion relation between sets of structures, or (information) states, where the premises hold and the set of states where the conclusion holds. Thus, in particular, standard

propositions describe sets of, or equivalently: *unary properties* over, states. And, the usual structural rules of standard logic reflect this picture: which makes them seem inevitable trivialities.

By contrast, much of the motivation for occurrence logics has a 'dynamic' ring: one wants to pay more attention to the actual processing of the information contained in the premises. At least to a first approximation, this might be modelled abstractly by making propositions *binary relations* over states (rather like the 'transition relations' associated with programs in computation). And then, an inference may be called valid if the initial and final state of any 'verification sequence' for the successive premises form a succesful transition pair for the conclusion also. Technically, this leads to the following interpretation (cf. van Benthem 1988*):

Let S be some arbitrary set of 'states'. Let R_a be any binary relation over S, for primitive types a. Next, inductively, let the various logical constants introduced in the above stand for suitable binary operations on relations over states. For instance, the Booleans have their obvious set-theoretic meaning, the concatenative product stands for relational composition, the Kleene star for (something very much like) transitive closure, etcetera. In particular, suitable clauses for the directed implications $\backslash$ and $/$ are possible too, witness

$$R_{a\backslash b} = \{ (x, y) \mid \textit{for all } (z, x) \in R_a : (z, y) \in R_b \} .$$

Then, valid inference for sequents $A \Rightarrow b$ may be defined in the above-mentioned way, for all their relational interpretations as described here.

It is easy to see that this provides a sound interpretation for the Lambek calculus (an observation due to Ewa Orlowska), as well as richer systems of linear logic based upon it. One natural issue remains, of course, whether this modelling is also *complete*.

But, even pending an answer to the latter question, it can be said that the relational interpretation provides some new insight. By a minimal modelling for some of the more dynamic aspects of interpretation, it explains at once why standard structural rules are neither trivial nor valid. To take the case of Contraction again: evidently, a composition $R_a \circ R_a$ need not be contained in R_a itself. Or alternatively, if one insists on contraction for some particular proposition, this will mean that its associated transition relation must be *transitive*. [It would be of interest, of course, to study various subclasses of propositions satisfying special mathematical properties.] But also, the proposed semantics explains the

proliferation of logical constants in occurrence logics: as we are now entering the area of Relational Calculus, which is known for its richer logical apparatus.

5.3 Variable-Free Notations

The semantic treatment in this paper leans heavily on a lambda calculus with several restrictions on its variable patterns. Recently, however, there has also been an emerging interest in *variable-free* formalisms, as being closer to the actual structure of natural languages. In this appendix, we shall consider a few such notations, in order to draw some comparisons.

The idea that natural language behaves in certain ways as if it were a variable-free companion of predicate logic has been put forward by many authors, such as Quine, Dummett and Geach. In particular, Geach 1972 contains a proposal for enriching categorial grammar - itself already a mechanism for doing away with some variables [for instance, EVERY(FOX)(LIES) replaces $\forall x(Fx \rightarrow Lx)$] - with a number of operators from Quine 1966, in order to deal with anaphors and reflexives. What Quine had invented (cf. the discussion of single-bond formalisms in Section 4) was a variable-free notation for predicate logic, based on operations which *identify* or *permute* arguments of predicates. Moreover, two standard conventions were introduced for the logical operators:

(1) quantifiers govern the last argument position of a subsequent predicate,
(2) connectives turn a subsequent m-ary and n-ary predicate into an (m+n)-ary one.

Then, a variable-free formalism arises which is intertranslatable with ordinary predicate-logical notation; and for which, e.g., intrinsic complete axiomatizations have been found in the meantime. (Cf. Bacon 1985. Some further discussion occurs in van Benthem 1988c.) Geach suggests that similar operators occur in natural language, relating identification to reflexives and permutation to passives.

But also, e.g., in intensional logic, similar viewpoints occur. After all, an operator formalism like that of propositional *temporal logic* is a kind of variable-free shorthand for avoiding explicit quantification over points in time - and a whole debate has raged concerning the propriety of this move as a reflection of temporal perspective in natural language (cf.

van Benthem 1977). The situation here was much clarified by the analysis given in Gabbay, Pnueli, Shelah and Stavi 1980, who showed that such operator formalisms are available for precisely those fragments of predicate logic, with some finite lexicon, which employ only up to some fixed finite number of distinct bound variables.

Remark: Actually, issues are a bit subtle here. For, after all, the above Quine formalism is a finite operator notation for all of predicate logic, without such bound variable restrictions. The crucial point, however, is that the Quine operators take predicates of arbitrary finite arities, rather than having some fixed categorial type. And, the G, P, S & S analysis only applies to the latter kind of operator. ♠

And finally, of course, there is *Combinatory Logic*, as an alternative formalism for the Lambda Calculus itself, which again provides variable-free notations for arbitrary lambda terms. For instance, the lambda terms of our earlier approach encode a Gentzen natural deduction format for categorial derivation - which also has an equivalent Hilbert-style axiomatic format, with corresponding combinatory terms.

Example: Combinators.
Sufficient combinators for the full lambda calculus are

$$S = \lambda xyz.\ xz(yz)$$

$$K = \lambda xy.\ x\ .$$

For the Lambek fragment (at least, as conceived in this paper), the following set suffices:

$$I = \lambda x.\ x$$

$$P = \lambda xy.\ yx$$

$$C = \lambda xyz.\ x(y(z))\ .$$

(With S added again, one gets the fragment allowing arbitrary non-vacuous lambda binding.) These combinators may be used to define others, such as our earlier reflexivizer with behaviour

$$Wfx = fxx\ .$$

For instance, here is a sentence which Geach 1972 translates into

the Quine formalism:

"Everyone who hurts everyone who hurts him, hurts himself"
$\forall x(\forall y\ (Hyx \rightarrow Hxy) \rightarrow Hxx)$.

In a standard categorial format, this would have the form

$\forall(\lambda x.\ \forall(\lambda y.\ H(y)(x), \lambda y.\ H(x)(y)), \lambda x.\ H(x)(x))$,

which can be translated into the following combinator expression:

$S(S(S(K\forall)(PH))H)(WH)$.

Although the Lambda Calculus formalism has been the main vehicle for fundamental logical research in the area (cf. Barendregt 1981), a combinator format has shown its virtues as well - especially in *computational* practice, where variable-free compilations of programs originally written in standard mathematical variable notation support more efficient execution. Steedman 1988 suggests that the syntax of natural language is in fact like such a variable-free compiled formalism. [In this case, the business of natural language *semantics* might be described as 'putting in variables' at strategic places.].

This situation raises a number of interesting questions, both semantic and computational. For instance, in which precise sense are variable-free formalisms computationally more tractable than variable-based ones? And, do they suggest some independent semantic hierarchy?

6. References

A. Avron, 1988,
'The Semantics and Proof Theory of Linear Logic',
Theoretical Computer Science 57, 161-184.

J. Bacon, 1985,
'The Completeness of a Predicate-Functor Logic',
Journal of Symbolic Logic 50, 903-926.

H. Barendregt, 1981,
The Lambda Calculus. Its Syntax and Semantics,
North-Holland, Amsterdam.

R. Bartsch, J. van Benthem and P. van Emde Boas, eds., 1989,

Semantics and Contextual Expression,
Foris, Dordrecht.

J. van Benthem, 1977,
'Tense Logic and Standard Logic',
Logique et Analyse 20, 41-83.

J. van Benthem, 1986,
Essays in Logical Semantics,
Reidel, Dordrecht.

J. van Benthem, 1987a,
'Categorial Grammar and Lambda Calculus',
in D. Skordev, ed., 1987, 39-60.

J. van Benthem, 1987b,
'Categorial Grammar and Type Theory',
To appear in *Linguistics and Philosophy.*

J. van Benthem, 1987c,
'Polyadic Quantifiers',
report 87-04, Institute for Language, Logic and Information, University of Amsterdam. (To appear in *Linguistics and Philosophy.*)

J. van Benthem, 1987d,
'Towards a Computational Semantics',
in P. Gardenfors, ed., 1987, 31-71.

J. van Benthem, 1988a,
A Manual of Intensional Logic,
second revised edition, Center for the Study of Language and Information, Stanford University / Chicago University Press.

J. van Benthem, 1988b,
'Logical Constants across Varying Types',
to appear in *Notre Dame Journal of Formal Logic.*

J. van Benthem, 1988c,
'Logical Syntax',
report 86-06, Institute for Language, Logic and Information, University of Amsterdam. (To appear in *Theoretical Linguistics.*)

J. van Benthem, 1988*,
'Semantic Parallels in Natural Languages and Computation',
report LP-88-06, Institute for Language, Logic and Information, University of Amsterdam. (To appear in M. Garrido, ed., 1989.)

J. van Benthem, 1988d,
'Strategies of Intensionalization',
in I. Bodnar et al., eds., 1988, 41-59.

J. van Benthem, 1988e,
'The Lambek Calculus',
in R. Oehrle et al., eds., 1988, 35-68.

J. van Benthem, 1988f,

'What is Extensionality?',
in I. Bodnar, ed., to appear.

J. van Benthem, 1989,
'Categorial Grammar Meets Unification',
in J. Wedekind, ed., to appear.

I. Bodnar, ed., to appear,
Annals Lorand Eotvos University.
Budapest, 1988.

I. Bodnar, A. Mate and L. Polos, eds., 1988,
A Filozofiai Figyelo Kiskonyvtara,
Kezirat Gyanant, Budapest.

W. Buszkowski, 1982,
Lambek's Categorial Grammars,
dissertation, Mathematical Institute, Adam Mickiewicz University, Poznan.

W. Buszkowski, 1988,
'Generative Power of Categorial Grammars',
in R. Oehrle et al., eds., 1988, 69-94.

D. Davidson and G. Harman, eds., 1972,
Semantics of Natural Language,
Reidel, Dordrecht.

J. van Eyck, 1985,
Aspects of Quantification in Natural Language,
dissertation, Filosofisch Instituut, University of Groningen.
(To appear with Reidel, Dordrecht.)

J-E Fenstad, P-K Halvorsen, T. Langholm and J. van Benthem, 1987,
Situations, Language and Logic,
Reidel, Dordrecht.

D. Gabbay & F. Guenthner, eds., 1984,
Handbook of Philosophical Logic, vol. II (Extensions of Classical Logic),
Reidel, Dordrecht.

D. Gabbay, A. Pnueli, S. Shelah and J. Stavi, 1980,
'On the Temporal Analysis of Fairness',
Proceedings 7th ACM Symposium on Principles of Programming Languages,
163-173.

D. Gallin, 1975,
Systems of Intensional and Higher-Order Modal Logic,
North-Holland, Amsterdam, (Mathematics Studies, vol. 19).

P. Gardenfors, ed., 1987,
Generalized Quantifiers. Linguistic and Logical Approaches,
Reidel, Dordrecht, (Studies in Linguistics and Philosophy, vol. 31).

M. Garrido, ed.,
Logic Colloquium. Granada 1987,
North-Holland, Amsterdam, (Studies in Logic).

P. Geach, 1972,
'A Program for Syntax',
in D. Davidon and G. Harman, eds., 1972, 483-497.

J-Y. Girard, 1987,
'Linear Logic',
Theoretical Computer Science 50, 1-102.

J. Groenendijk, D. de Jongh & M. Stokhof, eds., 1987,
Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers,
Foris, Dordrecht, (GRASS series, vol. 8).

D. Harel, 1984,
'Dynamic Logic',
in D. Gabbay & F. Guenthner, eds., 1984, 497-604.

H. Hendriks, 1988a,
'Generalized Generalized Quantifiers and Natural Natural Language',
Institute for Language, Logic and Information, University of Amsterdam.

H. Hendriks, 1988b,
'Type Change in Semantics: the Scope of Quantification and Coordination',
in E. Klein and J. van Benthem, eds., 1988, 96-119.

N. Immerman, 1982,
'Upper and Lower Bounds for First-Order Expressibility',
Journal of Computer and Systems Sciences 25:1, 76-98.

E. Keenan and L. Faltz, 1985,
Boolean Semantics for Natural Language,
Reidel, Dordrecht.

J. Ketting, 1988,
'Fuzzy Determinatoren en Quantificatie over Fuzzy Deelverzamelingen',
master's thesis, University of Amsterdam.

E. Klein and J. van Benthem, eds., 1988,
Categories, Polymorphism and Unification,
Centre for Cognitive Science (University of Edinburgh) /
Institute for Language, Logic and Information (University of Amsterdam).

Y. Lafont, 1988,
'The Linear Abstract Machine',
Theoretical Computer Science 59, 157-180.

J. Lambek, 1958,
'The Mathematics of Sentence Structure',
American Mathematical Annals 65, 154-169.

J. Lambek and P. Scott, 1986,
Introduction to Higher-Order Categorial Logic,
Cambridge University Press, Cambridge.

J. de Mey, 1988,
Dyadic Quantification,

dissertation, Instituut voor Algemene Taalwetenschap,
University of Groningen.

M. Moortgat, 1988,
Categorial Investigations. Logical and Linguistic Aspects of the Lambek Calculus,
dissertation, Faculteit der Wijsbegeerte, University of Amsterdam.

R. Muskens, 1988,
'Going Partial in Montague Grammar',
to appear in R. Bartsch et al., eds., 1989.

R. Oehrle, E. Bach and D. Wheeler, eds., 1988,
Categorial Grammars and Natural Language Structures,
Reidel, Dordrecht.

B. Partee, 1987,
'Noun Phrase Interpretation and Type Shifting Principles',
in J. Groenendijk et al., eds., 115-143.

G. Plotkin, 1980,
'Lambda Definability in the Full Type Hierarchy',
in J. Seldin and J. Hindley, eds., 1980, 363-373.

W. Quine, 1966,
'Variables Explained Away',
in *Selected Logic Papers*, Random House, New York.

J. Seldin and J. Hindley, eds., 1988,
To H. B. Curry. Essays on Combinatory Logic, Lambda calculus and Formalism,
Academic Press, New York.

D. Skordev, ed., 1987,
Mathematical Logic and its Applications,
Plenum Press, New York.

R. Statman, 1980,
'On the Existence of Closed Terms in the Typed Lambda Calculus. I',
in J. Seldin & J. Hindley, eds., 1980, 511-534.

R. Statman, 1982,
'Completeness, Invariance and λ-Definability',
Journal of Symbolic Logic 47, 17-26.

M. Steedman, 1988,
'Combinators and Grammars',
in R. Oehrle et al., eds., 1988, 417-442.

J. Wedekind, ed., to appear,
Proceedings Titisee Workshop on Unification Formalisms,
Institute for Computational Linguistics, University of Stuttgart.

L. Zadeh, 1988,
'Fuzzy Logic',
report CSLI-88-116, Center for the Study of Language and Information, Stanford University.

The ITLI Prepublication Series

1986

86-01 The Institute of Language, Logic and Information

86-02 Peter van Emde Boas — A Semantical Model for Integration and Modularization of Rules

86-03 Johan van Benthem — Categorial Grammar and Lambda Calculus

86-04 Reinhard Muskens — A Relational Formulation of the Theory of Types

86-05 Kenneth A. Bowen, Dick de Jongh — Some Complete Logics for Branched Time, Part I Well-founded Time, Forward looking Operators

86-06 Johan van Benthem — Logical Syntax

1987

87-01 Jeroen Groenendijk, Martin Stokhof — Type shifting Rules and the Semantics of Interrogatives

87-02 Renate Bartsch — Frame Representations and Discourse Representations

87-03 Jan Willem Klop, Roel de Vrijer — Unique Normal Forms for Lambda Calculus with Surjective Pairing

87-04 Johan van Benthem — Polyadic quantifiers

87-05 Víctor Sánchez Valencia — Traditional Logicians and de Morgan's Example

87-06 Eleonore Oversteegen — Temporal Adverbials in the Two Track Theory of Time

87-07 Johan van Benthem — Categorial Grammar and Type Theory

87-08 Renate Bartsch — The Construction of Properties under Perspectives

87-09 Herman Hendriks — Type Change in Semantics: The Scope of Quantification and Coordination

1988

Logic, Semantics and Philosophy of Language:

LP-88-01 Michiel van Lambalgen — Algorithmic Information Theory

LP-88-02 Yde Venema — Expressiveness and Completeness of an Interval Tense Logic

LP-88-03 — Year Report 1987

LP-88-04 Reinhard Muskens — Going partial in Montague Grammar

LP-88-05 Johan van Benthem — Logical Constants across Varying Types

LP-88-06 Johan van Benthem — Semantic Parallels in Natural Language and Computation

LP-88-07 Renate Bartsch — Tenses, Aspects, and their Scopes in Discourse

LP-88-08 Jeroen Groenendijk, Martin Stokhof — Context and Information in Dynamic Semantics

LP-88-09 Theo M.V. Janssen — A mathematical model for the CAT framework of Eurotra

LP-88-10 Anneke Kleppe — A Blissymbolics Translation Program

Mathematical Logic and Foundations:

ML-88-01 Jaap van Oosten — Lifschitz' Realizabiility

ML-88-02 M.D.G. Swaen — The Arithmetical Fragment of Martin Löf's Type Theories with weak Σ-elimination

ML-88-03 Dick de Jongh, Frank Veltman — Provability Logics for Relative Interpretability

ML-88-04 A.S. Troelstra — On the Early History of Intuitionistic Logic

ML-88-05 A.S. Troelstra — Remarks on Intuitionism and the Philosophy of Mathematics

Computation and Complexity Theory:

CT-88-01 Ming Li, Paul M.B.Vitanyi — Two Decades of Applied Kolmogorov Complexity

CT-88-02 Michiel H.M. Smid — General Lower Bounds for the Partitioning of Range Trees

CT-88-03 Michiel H.M. Smid, Mark H. Overmars Leen Torenvliet, Peter van Emde Boas — Maintaining Multiple Representations of Dynamic Data Structures

CT-88-04 Dick de Jongh, Lex Hendriks Gerard R. Renardel de Lavalette — Computations in Fragments of Intuitionistic Propositional Logic

CT-88-05 Peter van Emde Boas — Machine Models and Simulations (revised version)

CT-88-06 Michiel H.M. Smid — A Data Structure for the Union-find Problem having good Single-Operation Complexity

CT-88-07 Johan van Benthem — Time, Logic and Computation

CT-88-08 Michiel H.M. Smid, Mark H. Overmars Leen Torenvliet, Peter van Emde Boas — Multiple Representations of Dynamic Data Structures

CT-88-09 Theo M.V. Janssen — Towards a Universal Parsing Algorithm for Functional Grammar

CT-88-10 Edith Spaan, Leen Torenvliet Peter van Emde Boas — Nondeterminism, Fairness and a Fundamental Analogy

CT-88-11 Sieger van Denneheuvel Peter van Emde Boas — Towards implementing RL

Other prepublications:

X-88-01 Marc Jumelet — On Solovay's Completeness Theorem

1989

Logic, Semantics and Philosophy of Language:

LP-89-01 Johan van Benthem — The Fine-Structure of Categorial Semantics

Computation and Complexity Theory:

CT-89-01 Michiel H.M. Smid — Dynamic Deferred Data Structures