University of Massachusetts Occasional Papers in Linguistics

Volume 8 *Occasional Working Papers in Cognitive Science*

Article 5

1982

Montague's Intensional Logic and the Theory of Types

Raymond Turner

MONTAGUE'S INTENSIONAL LOGIC AND THE THEORY OF TYPES

Raymond Turner*

Is the type theory built into Montague's Intensional Logic (IL) a help or a hinderance? In the paper "The Proper Treatment of Quantification in Ordinary English" (commonly abbreviated PTQ) Intensional Logic is used as an intermediate specification language. English is translated into IL and the semantics of IL is separately provided. Given this task does IL perform it adequately? Furthermore, how well does it continue to perform this task when PTQ is extended in the way advocated by (for example) Bennett [1974]? The particular dimension we are interested in relates to the type theory, which forms a central component of IL. Does the type theory in IL serve any useful purpose or does it just get in the way of an elegant and intuitively correct semantic theory?

Why is IL based on the typed Lambda Calculus and not on Church's untyped system? I am sure that a full answer to this question should be partly historical but it is not in this aspect of the question that I am interested. Perhaps the question ought to be put in the following form: is there <u>now</u> any good reason why IL should be based on the typed as opposed to the untyped Lambda Calculus?

I believe that one argument we might try to construct, in favor of type-theory in IL, relates to the categorial structure of English syntax

---

* Supported in part by A. P. Sloan Foundation Grant 80-6-13.

University of Massachusetts Occasional Papers in Linguistics, Vol. 8 [1982], Art. 5

as it is presented in PTQ. Consider the syntactic categories IV (intransitive verb phrases) and IAV (intransitive adverbs). Expressions in IV denote functions in $E \rightarrow T$ (entities to truth-values) whereas expressions in IAV denote functions in $(E \rightarrow T) \rightarrow (E \rightarrow T)$. The whole of the Montague program is premised on the assumption that the meanings of expressions are functions of certain kinds. Once this assumption is made it is natural to employ higher-order functions of various kinds - as in the case of intransitive adverbs.

The next stage in the argument involves IL itself. Intensional Logic is introduced as a language intermediate between English surface structure and its semantic specification. If the meanings of various syntactic categories in English are ultimately to be various higher-order functions, and IL is to serve as an intermediary between English and its semantic specification, then IL must itself be a language capable of expressing such higher-order functions.

But what does this mean? Does it mean, for example, that our intermediate language needs to be typed? I do not think so. One can express a great deal more in an untyped Lambda calculus than one can in its typed counterpart. Certainly, if all we require is the ability to name higher-order functions then Church's untyped Lambda calculus is a reasonable candidate.

The purpose of this paper is not to examine all the arguments in favor of IL being typed. It has a much more positive purpose. We believe that type-theory may have a certain role to play in the semantics of natural language but that it is not best located in IL itself. As a consequence, we suggest a natural type-free alternative to IL and provide its syntax and semantics.

2. A PROBLEM FOR MONTAGUE'S INTENSIONAL LOGIC

I want to consider a problem which has recently been brought to our attention by Terry Parsons [1979]. I will illustrate the problem by reference to the following sentence:

(1) Jill is crazy.

According to the tradition of Montague grammar (ignoring intensionality) the meaning of a verb phrase like "is crazy" is a function from E (entities) to T (truth-values). Similarly, the meaning of the verb phrase "is writing papers" in (2) is a function from E to T:

(2) John is writing papers.

But what are we to make of the phrase "To write papers" in the sentence:

(3) To write papers is crazy?

Presumably, the meaning of "To write papers" also has functionality $E \to T$. But then, in (3), the verb phrase "is crazy" must have functionality $(E \to T) \to T$. We might continue in the like fashion. Consider the sentence:

(4) To be (so) crazy is beyond belief;

where the intention is to refer to the sense of "crazy" exemplified in (3). I take it, that in (4) the verb phrase "is beyond belief" has functionality $((E \to T) \to T) \to T$. In principle, given enough ingenuity and patience, we could continue this process indefinitely. But let's not.

We can best illustrate the problem more abstractly as follows. Consider the sequence of sentences:

(i) Every individual has a property;

(ii) Every property has a property;

(iii) Every property which some property has has a property;

(iv) Every property which some property which some property has has has a property;

etc.

In the first sentence the word "property will be represented semantically as a function in

$$P_0 = E \to T.$$

The "property" of sentence (ii) will have functionality

$$P_1 = P_0 \to T;$$

and generally properties of type n+1 will be functions from properties of type n to truth-values:

$$P_{n+1} = P_n \to T.$$

Where does "is crazy" come in this hierarchy? Apparently, everywhere; just like the word "property" the verb phrase "is crazy" denotes a function at all levels in the hierarchy. Unfortunately, IL itself cannot manage such complexity. In IL "is crazy" must be associated with a function of fixed type. Parsons seems to endorse the view that "is crazy" occurs everywhere throughout the hierarchy when he represents a verb phrase like "is crazy" as an infinite sequence of functions - one for each level in the hierarchy.

Indeed, as Parsons points out, such duplication of entities in one syntactic category leads to duplication in others. For example, the adverb "allegedly" is a verb phrase modifier and so has functionality Prop → Prop where Prop is the semantic domain of Properties. But, as we have just observed, Prop is a whole hierarchy of domains not just one. Subsequently, the meaning of "allegedly" is an infinite sequence of functions $\langle a_n \rangle_{n \in \omega}$ where $a_n \in M_n$ and where

$$M_0 = P_0 \to P_0;$$
$$M_{n+1} = M_n \to M_n.$$

So, on the face of it, we have a solution to the problem of sentences like 1 - 4 and i - iv. The word "property" is associated with an infinite sequence

of functions-one for each member of the hierarchy. But there is still something unsatisfactory about this solution. According to it the meaning of "is crazy" is an arbitrary sequence of functions $\langle f_n \rangle_{n \in \omega}$ where $f_n \in P_n$ for $n \in \omega$. But this is not all that our intuitions demand; they demand that f_n and f_{n+1} have something in common. We want to insist that when f_{n+1} is restricted to the domain of f_n then this function is exactly f_n. But as things stand we cannot do this since the domains P_n and P_{n-1} are disjoint. We need to make our function spaces cumulative in some way. Eventually, we shall achieve this with the aid of identification mappings between the domains.

We now turn our attention to sentences which seem problematic even for this theory. In his paper Parsons sets an exercise for the reader: try thinking to yourself "everything has some property" without restricting "everything" to things of a given type. I think this is an interesting challenge. I for one find it hard to think about this sentence without the structure imposed by type-theory. Type-theory seems able to play a conceptual role here and enables us to make sense of this sentence. So, at this stage in the analysis, I am very reluctant to give up type-theory. Nevertheless, we do seem to be able to assert

(5) Everything has a property

and mean every thing - not just things of some particular type. So in what sense do we do this? I believe that something like the following is going on. When we assert (5) we are asserting

(5n) Everything_n has a property_{n+1}

for every n. That is, we assert every thing of type_n has a property of type_{n+1} for every level in the hierarchy. Parsons has no way of saying this in his system. In fact, he quotes Russell's solution to this problem which is to supplement the grammar with a theory of pragmatics according to which

University of Massachusetts Occasional Papers in Linguistics, Vol. 8 [1982], Art. 5

when we assert the above sentence (5) we automatically assert all of its meanings i.e. for each n, one asserts that everything of type n has a property. Whether this is part of the semantics or the pragmatics is a moot point but how does one get a formal theory to capture the intuition in either case?

Similar considerations seem to apply to the sentence:

(6) Being crazy is (just) crazy.

I think there is a clear sense to this sentence but how are we to interpret it in the context of our hierarchy? Presumably, we have made a statement in form similar to (5); when one asserts (6) one automatically asserts all of its meanings i.e. for each n, one asserts:

(6n) Being crazy_n is (just) crazy_{n+1}.

In the context of type theory, I can see no other way to make sense of the English sentence (6). Note that Parsom's theory would supply no meaning at all to (6) since, presumably, the type of "is" (or "has") would rule it ungrammatical.

One thing seems clear. There is no hope of capturing these intuitions in a language like IL which is typed. In such a language verb phrases like "is crazy" will be assigned a specific type. English is just not a typed language in this strong sense. In section 5 we introduce a language (which might fulfill this role of an intermediate language) which is not typed,

3. FUNCTIONS OF INFINITE TYPE

These considerations seem to impose two constraints or adequacy conditions on any intuitively correct semantic theory:

(I) We need to be able to represent the meanings of words and phrases (like "property", "is crazy", "allegedly") as infinite sequences of functions which satisfy certain conditions;

(II) We require a definition of application (and indeed self-applications) for such infinite sequences.

Fortunately, Dana Scott has provided us with a ready-made mathematical theory. We now outline the main ideas behind Scott's theory. The treatment will be very brief and only enough detail will be included to whet the readers appetite. The full details are available in Scott [1972] and Barandregt [1977].

The following two notions are fundamental:

COMPLETE LATTICES

A complete lattice is a partially ordered set $\langle D, \sqsubseteq \rangle$ such that each subset $X \subset D$ has at least upper bound.

(Notice that this guarantees the existence of greatest lower bounds as well as the existence of a greatest element or top ($\top$) and a least element or bottom ($\bot$)).

CONTINUOUS FUNCTIONS

Let D, D' be complete lattices. A function $f : D \to D'$ is continuous iff for each directed set $X \subset D$, $f(\bigsqcup X) = \bigsqcup \{f(x) : x \in X\}$ where X is directed in D iff $(\forall x, y \in X)(\exists z \in X)(x, y \sqsubseteq z)$.

Given these two ideas we can build new lattices from old ones. The following three constructions are the crucial ones for the application we have in mind:

SUM CONSTRUCTIONS

Let D, D' be complete lattices. The Sum $D+D'$ is the disjoint union of D and D' with the addition of new elements $\perp, \top$ such that, for each $x \in D$ or $x \in D'$, $\perp \sqsubseteq x \sqsubseteq \top$. $D+D'$ is a complete lattice.

PRODUCT CONSTRUCTION

Let D, D' be complete lattices. The product $D \times D'$ is the space $D \times D' = \{\langle x,y\rangle : x \in D \text{ and } y \in D'\}$ where $\langle x,y\rangle \sqsubseteq \langle x',y'\rangle$ iff $x \sqsubseteq x'$ and $y \sqsubseteq y'$ for $x,x' \in D$ and $y,y' \in D'$. $D \times D'$ is a complete lattice.

The next construction is the important one. It informs us how we are to turn the space of continuous functions (from one complete lattice into a second) into a complete lattice

FUNCTION SPACE CONSTRUCTION

Let D, D' be complete lattices. The space of continuous functions from D into D' (written $[D \to D']$) consists of all continuous functions from D into D; where

$$f \sqsubseteq g \iff (\forall d \in D)(f(d) \sqsubseteq' g(d)).$$

Least upper bounds are computed by

$$\sqcup F = \lambda d \cdot \sqcup\{f(d) : f \in F\}$$

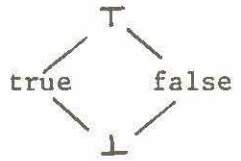
for $F \subseteq [D \to D']$. The structure $[D \to D']$ is a complete lattice.

This completes the basic mathematical background. Our intension is to iterate the function space construction in the same way as we constructed the function spaces of properties - the only difference being we restrict attention to <u>continuous</u> functions at each level. We begin with a domain (lattice) which is largely motivated by our preliminary discussion. Let

$$V_0 = E + Bool$$

where E is some basic domain (called individuals in PTQ perhaps) and Bool is the domain of Boolean values:

Bool =



.

We then define

$$V_{n+1} = [V_n \to V_n] \quad \text{(continuous functions)}.$$

Our first constraint demands that we make these function spaces cumulative. We do this by defining a sequence of continuous functions (projections):

$$\phi_n : V_n \to V_{n+1}$$

$$\psi_n : V_{n+1} \to V_n$$

for each $n \geq 0$. We define this sequence by induction on n:

$$\phi_0(x) = \lambda y \in V_0 \cdot x$$

$$\psi_0(x') = x'(\perp) \quad ;$$

and

$$\phi_{n+1}(x) = \phi_n \circ x \circ \psi_n$$

$$\psi_{n+1}(x') = \psi_n \circ x' \circ \phi_n,$$

Note that in higher types it would be unreasonable to use the constant function identification we used for ϕ_0, since such a mapping would destroy the functional character of these objects. These functions satisfy: $\psi_n(\phi_n(x)) = x$ and $\phi_n(\psi_n(x')) \sqsubseteq x'$; this can be established by induction on n. In fact we can extend the mappings ϕ_n, ψ_n to mappings $\phi_{nm} : V_n \to V_m$ as follows

$$\phi_{nm}(x) = \begin{cases} \phi_{m-1} \circ \cdots \circ \phi_n & n < m \\ x & n = m \\ \psi_m \circ \cdots \circ \psi_{n-1} & m < n. \end{cases}$$

Once again we leave the reader to check that $\phi_{mn}(\phi_{nm}(d)) = d$ and $\phi_{nm}(\phi_{mn}(d')) \sqsubseteq d'$ for $0 \leq n \leq m$.

We can now define our domain of <u>functions of infinite types</u>. According to our intuitions it is to be constituted of infinite sequences of functions $\langle f_n \rangle_{n \in \omega}$ which are related by the condition that f_n is the best "approximation" to f_{n-1} available at level n - in terms of our mappings this means $\psi_n(f_{n+1}) = f_n$.

V_∞-CONSTRUCTION

The domain V_∞ is the set

$$V_\infty = \{\langle x_n \rangle_{n\in\omega} : (\forall n)(\psi_n(x_{n+1}) = x_n)\}$$

where for $\langle x_n \rangle_{n\in\omega}$, $\langle y_n \rangle_{n\in\omega}$ we define

$$\langle x_n \rangle_{n\in\omega} \sqsubseteq \langle y_n \rangle_{n\in\omega} \iff (\forall n)(x_n \sqsubseteq y_n).$$

V_∞ is a complete lattice.

We can now define

$$\phi_{n\infty} : V_n \to V_\infty \quad \text{and}$$

$$\phi_{\infty n} : V_\infty \to V_n$$

by

$$\phi_{n\infty}(x) = \langle \phi_{ni}(x) \rangle_{i\in\omega} \quad \text{and}$$

$$\phi_{\infty n}(x) = x_n.$$

Once again we leave to the reader the task of proving $\phi_{\infty n}(\phi_{n\infty}(x)) = x$ and $\phi_{n\infty}(\phi_{\infty n}(x')) \sqsubseteq x'$.

It follows that, up to isomorphism,

$$V_0 \subseteq V_1 \subseteq V_2 \subseteq V_3 \subseteq \ldots V_\infty.$$

It is this identification which enables us to view is "is crazy$_n$" as an approximation of "is crazy$_{n+1}$".

One of our objectives has been fulfilled. We have shown how to define a mathematical structure whose elements are infinite sequences of functions which conform to our constraints.

We can now turn to the second requirement of our theory: how are we to apply such sequences to each other?

APPLICATION IN V_∞

Let $x,y \in V_\infty$. Then define

$$x \circ y = \bigsqcup_{n\in\omega} x_{n+1}(y_n).$$

The reader should perhaps check (or look-up) that application is continuous and satisfies $x_{n+1} \circ y = x_{n+1} \circ y_n = (x \circ y_n)_n$.

But what has become of type-theory in all this? This is an important question given our belief that type theory plays some role in our understanding of sentences like (5) and (6). If you recall this belief amounts to the view that when we assert (5) we assert (5n) for each n. How does our definition of application fare here? The word "property" and "thing" get associated with elements of V_∞. To assert the sentence (5) is equivalent to

$$(\forall f \exists g)\ (g(f) = \text{true});$$

which in V_∞ amounts to

$$(\forall f)(\exists g)((\bigsqcup_{n\in\omega} g_{n+1}(f_n)) = \text{true}).$$

This amounts to the claim that, for each n, either $g_{n+1}(f_n) = \text{true}$ or it is undefined ($=\perp$) - and it is defined somewhere. This seems to be what our intuitions demand.

This all seems rather satisfactory. We have a theory which conforms to our intuitions regarding sentences like (1) - (6) and we have what seems to be an appropriate role for type-theory in the formal semantics of natural language. This brings us to the main theorems of Scott (1972).

Theorem. (completeness)

$$(\forall f \in [V_\infty \to V_\infty])(\exists x \in V_\infty)(\forall y \in V_\infty)(f(y) = x \circ y),$$

In fact, the x required is $x = \bigsqcup_{n\in\omega} (\lambda y \in V_n \bullet (f(y)_n))$. The Theorem follows by a relatively straightforward computation for $x \circ y$.

Theorem. (Isomorphism theorem)

V_∞ is isomorphic to $[V_\infty \to V_\infty]$.

The isomorphism is given by the function ϕ where

$$\phi(x) = \lambda y \in V_\infty \bullet (x \circ y);$$

ϕ is a one-one and onto continuous mapping $V_\infty \to [V_\infty \to V_\infty]$. Thus, up to isomorphism, V_∞ and $[V_\infty \to V_\infty]$ are the same complete lattice and we shall often indicate this by using the notation

$$V_\infty = [V_\infty \to V_\infty],$$

The technique employed in the construction of V_∞ is a perfectly general one. It can be used to solve any "system of equations" involving "+", "×" and "→". The following system for example, occurs in the next section

$$\begin{aligned} V &= E + BOOL + FUN + SINN \\ SINN &= [INDEX \to V] \\ FUN &= [V \to V] \\ INDEX &= W \times T. \end{aligned}$$

The most elegant context in which to carry this out is category theory cf. Smith & Plotkin [1977].

4. NOMINALIZATION

We might have reacted to sentences like (1) and (6) in a rather different way. Consider some of these once more:

(1) Jill is crazy

(2) John is writing papers

(3) To write papers is crazy (to type papers is crazier)

(4) Being crazy is (just) crazy.

Our initial persuasion was to react to the multiplicity of "is crazy"'s by claiming that the verb-phrase "is crazy" denotes an infinite sequence of functions. But there is a different way to proceed; a way which amounts to the claim that "is crazy" really denotes just one function.

Our discussion in the previous section guarantees the existence of a domain OBJ which satisfies the following equation:

$$OBJ = [(E + OBJ) \rightarrow BOOL],$$

where E is some basic domain (what PTQ calls individuals). To put the matter more precisely we are guaranteed the existence of homeomorphisms ϕ and ψ:

$$\phi : OBJ \rightarrow [(E + OBJ) \rightarrow BOOL]$$

$$\psi : [(E + OBJ) \rightarrow BOOL] \rightarrow OBJ.$$

For the sake of clarity we shall call the domain $E + OBJ$ the domain of things (THINGS say). Let's reexamine (1)-(3) and (6) with these considerations in mind. In sentences (1) and (2) the verb-phrases "is crazy" and "is writing papers" denote functions in $THINGS \rightarrow BOOL$. In sentence (3) "To write papers" is really a nominalized form of the verb phrase "is writing papers". The homeomorphism ψ is the one which we shall interpret as the process of nominalization. To see what is involved in (3)

let f denote the meaning of "is writing papers" i.e. f is an element of $[(E+OBJ) \rightarrow BOOL]$ (or $[THINGS \rightarrow BOOL]$). The meaning of "to write papers" is then given by the nominalization mapping as $\psi(f) \in OBJ$ - which is a subdomain of THING (by definition). Everything now works since f, being an element of $[THINGS \rightarrow BOOL]$, can be an applied to $\psi(f) \in THINGS$. Finally, we examine sectence (6). Once again "Being crazy" is a nominalized form of "is crazy" and so, if f is the function which represents the meaning of "is crazy", then $\psi(f)$ will represent the meaning of "Being crazy". Since $\psi(f) \in THINGS$ and $f \in [THINGS \rightarrow BOOL]$ it makes perfect sense to apply f to $\psi(f)$.

We have thus provided an analysis of (1)-(3) and (6) without any explicit appeal to type-theory. It, is of course, present in the background in as much as type-theoretic ideas have been used in the construction of these domains (e.g., OBJ). Nevertheless, once the existence of the domains is guaranteed we can provide a satisfactory account of these sentences without type-theory playing any explicit role in the analysis. We have climbed up the ladder of types to arrive at a point where the ladder can be dispensed with.

So perhaps my initial claim, that sentences like (5) and (6) could not be understood or made sense of without some appeal to type-theory, was too hasty. Perhaps, the proper role of type-theory is not to be located in the analysis of sentences like (5) and (6) but rather in guaranteeing the existence of domains like OBJ. What ever interpretation of (5) or (6) you prefer, however, our theory can handle it.

There is more to say about this interpretation of nominalization especially how it relates to Frege's views on the subject. This will be saved for a different occasion.

5. A TYPE-FREE VERSION OF INTENSIONAL LOGIC

In this section we propose an "untyped"version of IL. We provide its syntax and semantics. The language we propose (called "EVIL" for reasons which are best left unsaid) is similar to IL but the Lambda calculus component is based on Church's untyped Lambda calculus.

A. Syntax of EVIL

The basic expressions consist of a denumerably infinte set of constants (non logical) $c \in C$ and a denumerably infinite set of variables $x \in X$. The meaningful expressions of EVIL are defined recursively as follows:

1. Every variable and constant (non-logical) is in EVIL.
2. If E_1 and E_2 are (in) EVIL so is $\sim E_1$, $E_1 \wedge E_2$, $E_1 = E_2$.
3. If x is a variable and E is (in) EVIL then so is $\exists x\, E$ and $\lambda x.\, E$
4. If E_1 and E_2 are EVIL then so is $E_1(E_2)$.
5. If E is EVIL then so is $\Box E$
6. If E is EVIL then so is PE and FE.
7. If E is EVIL then so is $^{\wedge}E$ and $^{\vee}E$.
8. Nothing else is EVIL

B. Semantics of EVIL

First we need to know what a model of EVIL looks like. We shall restrict attention to those domains which are complete lattices. So, in particular, we can solve recursive domain equations; such equations will form an integral part of our models. A model for EVIL will be an ordered six-tuple.

$$\mathcal{M} = \langle E, V, W, T, <, F \rangle$$

where

(i) $F : C \to V$;

(ii) W, T are non-empty domains;

(iii) "<" is a linear-ordering of T which is continuous;

(iv) E is a non-empty domain (basic values);

(v) V is a non-empty domain.

In addition, we demand that our domains satisfy the following equations:

$$V = E + BOOL + FUN + SINN$$
$$SINN = [INDEX \to V]$$
$$FUN = [V \to V]$$
$$INDEX = W \times T$$

where BOOL is the domain of truth-values; SINN the domain of senses and FUN the domain of functions.

To provide the semantic function itself we require one further domain to provide the values of variables:

$$g \in ASG = [X \to T]$$

The domain ASG is the domain of assignments. The assignment $g[x|v]$ where $x \in VAR$ and $v \in V$ is:

$$g[x|v] = \lambda y \cdot \begin{cases} v & x = y \\ g(y) & \text{otherwise} \end{cases}$$

C. Semantic Function

We define a function

$$VAL : EVIL \to [ASG \to [W \times T \to V]]$$

by recursion on the structure of EVIL. We shall assume the model $\mathcal{M}$ is fixed and leave out all reference to it in what follows.

(v1) $VAL[\![c]\!]_{gwt} = F(c)$

(v2) $VAL[\![x]\!]_{gwt} = g(x)$

(v3) $VAL[\![\alpha \wedge \beta]\!]_{gwt} = \begin{cases} \text{true} & \text{if } VAL[\![\alpha]\!]_{gwt} = \text{true} \\ & \text{and } VAL[\![\beta]\!]_{gwt} = \text{true} \\ \text{false} & \text{if } VAL[\![\alpha]\!]_{gwt} = \text{false} \\ & \text{or } VAL[\![\beta]\!]_{gwt} = \text{false} \\ \perp & \text{otherwise} \end{cases}$

(v4) $VAL[\![\sim\alpha]\!]_{gwt} = \begin{cases} \text{true} & \text{if } VAL[\![\alpha]\!]_{gwt} = \text{false} \\ \text{false} & \text{if } VAL[\![\beta]\!]_{gwt} = \text{true} \\ \perp & \text{otherwise} \end{cases}$

(v5) $VAL[\![\alpha = \beta]\!]_{gwt} = \text{true}$ iff $VAL[\![\alpha]\!]_{gwt} = VAL[\![\beta]\!]_{gwt}$ (false otherwise)

(v6) $VAL[\![\exists x\, \alpha]\!]_{gwt} = \begin{cases} \text{true} & \text{if there exists a } v \in V \text{ such} \\ & \text{that } VAL[\![\alpha]\!]_{g[x|v]wt} = \text{true} \\ \text{false} & \text{if for each } v \in V \\ & VAL[\![\alpha]\!]_{g[x|v]wt} = \text{false} \\ \perp & \text{otherwise} \end{cases}$

(v7) $VAL[\![\lambda x \cdot \alpha]\!]_{gwt} = \lambda v \in V \cdot VAL[\![\alpha]\!]_{g[x|v]wt}$

(v8) $VAL[\![\alpha(\beta)]\!]_{gwt} = VAL[\![\alpha]\!]_{gwt} \upharpoonright FUN(VAL[\![\beta]\!]_{gwt})$

where

$$v \upharpoonright FUN = \lambda a \cdot \begin{cases} v(a) & \text{if } v \in FUN \\ \perp & \text{otherwise} \end{cases}$$

$$\text{(v9)}\quad \mathrm{VAL}[\![\Box\alpha]\!]_{gwt} = \begin{cases} \text{true} & \text{if for each } w' \in W \\ & \mathrm{VAL}[\![\alpha]\!]_{gw't} = \text{true} \\ \text{false} & \text{if there exists } w' \in W \text{ st.} \\ & \mathrm{VAL}[\![\alpha]\!]_{gw't} = \text{false} \\ \bot & \text{otherwise} \end{cases}$$

$$\text{(v10)}\quad \mathrm{VAL}[\![P\alpha]\!]_{gwt} = \begin{cases} \text{true} & \text{if there exists } t' < t \text{ st} \\ & \mathrm{VAL}[\![\alpha]\!]_{gwt'} = \text{true} \\ \text{false} & \text{if for each } t' < t \\ & \mathrm{VAL}[\![\alpha]\!]_{gwt'} = \text{false} \\ \bot & \text{otherwise} \end{cases}$$

$$\text{(v11)}\quad \mathrm{VAL}[\![F\alpha]\!]_{gwt} = \begin{cases} \text{true} & \text{if there exists } t' > t \text{ st} \\ & \mathrm{VAL}[\![\alpha]\!]_{gwt'} = \text{true} \\ \text{false} & \text{if for each } t' > t \\ & \mathrm{VAL}[\![\alpha]\!]_{gwt'} = \text{false} \\ \bot & \text{otherwise} \end{cases}$$

$$\text{(v12)}\quad \mathrm{VAL}[\![\hat{}\alpha]\!]_{gwt} = \lambda w' \lambda t'\ \mathrm{VAL}[\![\alpha]\!]_{gw't'}$$

$$\text{(v13)}\quad \mathrm{VAL}[\![\check{}\alpha]\!]_{gwt} = (\mathrm{VAL}[\![\alpha]\!]_{gwt} \restriction \mathrm{SINN}.)wt$$

A little discussion of EVIL is clearly in order. Most of the clauses parallel those of IL except that certain expressions are undefined. The clause (v13) is of some interest. Notice how the operation "$\restriction$" restricts one to elements of SINN. This replaces the purely syntactic constraint imposed by IL.

As a side remark note that Parsons example about "the property of not exemplifying itself" does not create problems for us. To see this let

$$\mathcal{P} = \lambda x \cdot \sim x(x)$$

be the property of not exemplifying itself. Then if we evaluate $VAL[\![\mathcal{P}(\mathcal{P})]\!]$ we discover this to be equal to (look at the definition of VAL). $VAL[\![\sim\mathcal{P}(\mathcal{P})]\!]$. This can only be so if both are equal to $\perp \in BOOL$.

We leave a more detailed investigation of EVIL for another occasion* but certainly the translation of the (PTQ) subset of English into EVIL would be identical to that for IL. Of course, one can do a lot more. One can, for example, handle all of Parson's extension and I believe all the material in Chierchia [1981]. I recommend EVIL to the Montague Grammarians.

*In particular one has to prove the following: for each ϕ in EVIL the function

$$VAL[\![\phi]\!] : ASG \to [WXT \to V]$$

is continuous in each of its variables. This is straightforward (induction on ϕ) but tedious. We leave it to the reader -- indulge yourself in a little EVIL.

Bibliography

Barandregt, H. (1977): "The Type-Free Calculus"; Handbook of Mathematical Logic 1977, North Holland. (A good clear introduction to Scott's theory.)

Bennett, M. (1974): "Some Extensions of a Montague Fragment of English"; Ph.D. UCLA, 1974.

Chierchia, G. (1981): "Nominalization and Montague Grammar" (UMass at Amherst) (This paper was part of the inspiration for this work. Chierchia uses a system of second-order logic due to Cocchiarella to replace IL. It is a rich source of linguistic examples and provides further motivation for what has been done here. In particular, there is a detailed discussion of nominalization.)

Cresswell, M. (1973): Logics and Languages: Methuen. (Contains an alternative approach to ours using partial functions.)

Montague, R. (1974): "Formal Philosophy" Yale. Ed. R. Thomason.

Parsons, T. (1979): "Type Theory and Ordinary Language" in Linguistics, Philosophy and Montague Grammar, Texas, 1979. Eds. Davis and Methuen (Contains the details of the theory alluded to in this paper.)

Scott, D. (1972): "Continuous Lattices", in: Toposes, Algebraic Geometry and Logic, Ed. F.W. Lawrence, LN Maths. Vol. 274, Springer. (The complete theory but uses topological ideas).

Russell, B. (1903): "The Principles of Mathematics" Allen & Irwin.

Smyth & Plotkin (1977): "The category-theoretic solution of recursive domain equation Proc. 18th - Annual IEEE Symposium on Foundations of Computer Science, 1977. (A category-theoretic account of Scott's theory)

Scott, D. (1978): "Data Types as Lattices": SIAM Journal of Computing, Vol. 5, No. 3, 1976. (The best source for a clear introduction to the whole theory.)

112.