Tertium Non Datur **3** (1986): 241–255

Logical types and linguistic types

András Kornai
Institute of Linguistics, Hungarian Academy of Sciences

0 Introduction

One of the primary aims of linguistic semantics is to translate the expressions of natural language into formulas of some logical calculus. These formulas, in turn, can be interpreted in the appropriate models, and semantic notions like truth, entailment, etc. can be formally defined in the usual manner. In addition to the well-known theoretical advantages of such an intermediate logical form, there is a practical advantage as well: given a system of rules for translation from formulas to natural language, it will be possible to translate from one natural language to another (via the interlingua) without actually evaluating the expressions of the source language. Although the calculi used as intermediate language in machine translation range from first-order predicate calculus (e.g. Schubert – Pelletier 1982) to the higher order intensional calculus of Montague Grammar (e.g. Landsbergen 1977), so far no type-free calculus has been employed for this purpose.

The aim of this paper is to outline an intermediate language for machine translation which is based on combinatory logic. Section 1 sketches the background assumptions of type theory. Section 2 discusses some of the problems with the traditional (typed) approach in Montague Grammar, and Section 3 outlines some general problems with type theory. In Section 4 a type-free intermediate language is defined and exemplified: its interpretation is discussed in the last section.

1 Naive type theory

The naive theory of semantic types is based on the assumption that the world is a collection of 'things': these will correspond to the individual constants of the model. Some of these things are animate (and thus can act upon others), and all of them can be individuated, so it can be always known which is which. Although only a few (and usually only animate) individuals have names, the common-sense assumption is that we can give a name to every thing; thus, *proper names* will have the type e.

Common nouns are usually taken to represent collections of things: therefore, they are interpreted as sets of individuals and will have the type $(e \rightarrow t)$. It is less clear what to do with abstract nouns like *love*, and the decision here is sometimes based on the linguistic similarity of abstract and concrete nouns (so abstract nouns are also taken to be of type $(e \rightarrow t)$), and sometimes on the common sense observation that love is between individuals, so it "must be" a relation (of type $e \times e$). What we see here is but the first conflict between the common sense guided

by linguistic intuition and the (equally) common sense guided by 'folk ontology' – most of the problems discussed here are the result of some similar conflict.

There is little doubt that adjectives refer to properties of things: therefore, the most natural choice for their type is $(e \to t)$. In the classical grammatical tradition, we might even define nouns with the aid of this particular type: the linguistic expression for a thing is a 'noun substantive', and for a property a 'noun adjective'. For reasons to be discussed in 2 below, more sophisticated type theories usually do not assign the same type to nouns and adjectives (except for predicate adjectives), but here we are concerned only with the naive theory of types. Intransitive verbs are just like adjectives (*to walk* is to have the property *is walking*). This can also be seen from the fact that the application of an intransitive verb (e.g. *walks*) to an individual (e.g. *John*) gives a full sentence which "must be" of type t: on both accounts, intransitive verbs are of type $(e \to t)$. There seems to be no way to extend the naive approach to transitive (ditransitive, etc) verbs: folk ontology considers these to be 'actions', and actions belong to a fundamentally different (dynamic) type. Actions like writing or killing can create new entities and destroy old ones: this makes it particularly hard to interpret them, even as higher order functions, in a (naive) set theoretical model.

2 Montague Grammar

The type theory of Montague (1970a, henceforth UG) is essentially the same as the naive theory outlined above; adjectives, however, have the type $(e \to t) \to (e \to t)$, since (as it was pointed out e.g. by Parsons 1970) their denotation might depend on the noun they modify.[1] This solution is also in accordance with the observation that adjectives combine with nouns into phrases syntactically equivalent to nouns (using the notations of categorial grammar, A = N/N), so if adjectives take $(e \to t)$ elements as their arguments, their values must also be of type $(e \to t)$.[2] In general, if x is of category A/B, the type of A is u, and the type of B is v, then x must have type $(v \to u)$. I will call this requirement the *soundness* of type-assignment (for a more formal statement, see 3 below). Although considerations of soundness play an important role in type-assignment in Montague Grammar (especially for higher types), it should be kept in mind that the above requirement is insufficient for unique type-assignment unless we stipulate that the only permitted mode of composition is *application*[3] (of a function to an argument), and we know which constituent is the function and which is the argument. The importance of this latter condition is best exemplified by the radically different types assigned to proper

[1] To use the original examples, a *married man* is a man having the property *married*, but a *big flea* is not a flea having the (absolute) property *big*, and a *former president* is not even a president.

[2] To facilitate comparison with the naive theory, the intensional types of Montague Grammar are systematically replaced by their extensional variants.

[3] This stipulation is not necessarily a part of Montague's original program (as formalized in UG), but it is implicit in the work of most linguists in the Montague tradition. For an explicit statement, see Hausser 1982 ch 1.

names in the earlier work of Montague and in later treatments (Montague 1973, henceforth PTQ). Proper names (type e in Montague 1970b) and intransitive verbs (type $e \rightarrow t$) combine to form sentences (type t). Therefore, if the verb is the function, it must be of type $(e \rightarrow t)$ for the type assignment to be sound, but if we take the noun to be the function (as in PTQ), the only way to maintain the type $(e \rightarrow t)$ for verbs and the type t for sentences is to assign the type $(e \rightarrow t) \rightarrow t$ to proper nouns.

It is questionable whether this type-assignment has any intuitive appeal, and in any case, we have to introduce meaning postulates (as in PTQ) or to enrich Russellian type theory considerably (as in Keenan 1981) in order to develop a workable system.

But the main problem is that no version of Montague Grammar has a one-to-one correspondence between logical types and linguistic types. For instance, common nouns and intransitive verbs, though clearly different linguistically, have the same logical type $(e \rightarrow t)$ in every theory mentioned so far. The distinction between t/e and $t//e$ was made in PTQ precisely in order to cope with this problem, and in more detailed systems (such as Partee 1977) the use of three, four, or even more slashes is quite common. This makes it impossible to define a function from logical types to linguistic types, and Williams (1983) argues that no function can be defined in the opposite direction either.

3 Problems with types in general

In order to provide a formal definition of linguistic categories, it will be necessary to treat natural languages as stringsets, i.e. formal languages. The *terminal vocabulary* V will contain the morphemes of the language, and $L \subseteq V^*$ is given by those strings of morphemes which (when entered to the phonological component of the grammar as input) give rise to grammatical sentences of the language in question. If we define the syntactic congruence $=$ in the usual manner, *lexical categories* will be simply the congruence classes of V. [4] The *syntactic categories* of L will be the congruence classes of V^* – some of these will be 'lexical categories' (if they contain strings of length one) but there might be others that have no lexical representatives.[5] In general, a *type-assignment* is simply a function f from V* to some set t of type symbols. Usually, T has some internal structure: for instance, the set R of *Russellian types* is generated from the atomic type symbols e and t by the rules

(i) e and t are in T
(ii) if $a_1 a_2 \ldots a_n$ are in R then $(a1 \rightarrow a2)$ and $(a_1 \times a_2 \times \ldots \times a_n)$ are in R
(iii) there are no other elements in R

[4] $a = b$ iff for every c, d in V^* *cad* is in L just in case *cbd* is in L, that is, $a = b$ iff they have the same distribution. These definitions are implicit in the early structuralist literature: for a detailed statement, see Harris (1951 ch 15).

[5] In the X-bar theory of syntactic categories (Harris 1951, Jackendoff 1977) it is sometimes assumed that every constituent category is lexical. In purely distributional terms this means that every constituent can be substituted by some lexical item.

A type assignment f will be called *syntactic* if it is compatible with the syntactic congruence: if rules of translation are required to be uniform for members of the same syntactic category (as they usually are), f will necessarily be syntactic. (The argumentation in Williams (1983) purports to show that no type assignment can be syntactic – but the concept of syntactic category employed there is broader than the definition used here.)

The *soundness* of f means that f is a homomorphism from the syntactic monoid $V^*/=$ to T.[6] Under this definition, only syntactic type-assignments can be sound. Finally, a translation will be *saturated* if for every type either no relation of that type is the interpretation of some expression, or every relation can be an interpretation.

Turner (1983) uses the process of nominalization to show that if we translate English into formulas interpreted in Russellian typed models, than the saturatedness of the translation and the soundness of the associated type-assignment are incompatible. A similar argument can be based on the Hungarian suffixes *-ít* and *-ó*. *-ít* can be used productively to form transitive verbs (type $f(N) \times f(IV)$) from adjective (type $f(A)$), and *-ó* can be used productively to form adjectives from transitive (and also from intransitive) verbs. Therefore, *-ít* establishes a one-to-one correspondence between adjectives and a subset of transitive verbs, and *-ó* establishes a one-to-one correspondence between adjectives and a subset of transitive verbs. Since the syntactic rules associated with these suffixation processes are fully regular (and compositional), the same correspondences must hold among verb and adjective translations as well. Therefore, by the Cantor - Bernstein theorem, the cardinality of type $f(N) \times f(IV)$ relations must be the same as that of type $f(A)$ relations in any model, and this is impossible if the fundamental set (of type e entities) is finite.

Since category-changing affixes of the above sort are present in most (perhaps all) natural languages, either the soundness of type-assignment or the saturatedness of the translation must be violated if we interpret our formulas in the usual Russellian models. To resolve this problem, Keenan (1983) develops a weaker notion of saturatedness (but retains the usual models), while Turner (1983) retains saturatedness (but uses Scott domains as models). My proposal is to weaken the requirement of soundness so that it will be satisfied trivially: this will be achieved by taking the monoid T to be a one-member set.

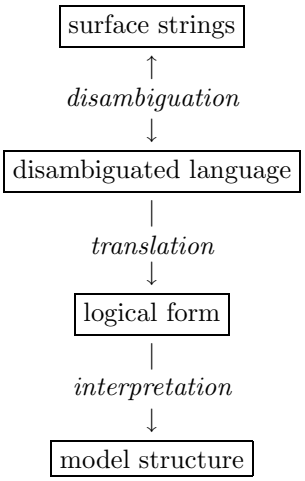
4 The intermediate language

Readers conversant in Montague Grammar will have noticed that the above presentation made no use of the ambiguation relation of UG: in other words, the disambiguated language and the surface strings of natural language were taken to be identical (modulo morphology). Although heavy use of (dis)ambiguation would make it possible to reformulate almost every grammatical theory in UG terms (cf.

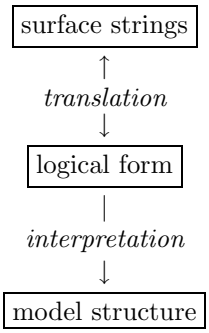
[6] The monoid structure on R can be defined trivially with the adjunction of a single element (which will be the product of those types that cannot be composed) – in general, T is supposed to have some 'natural' monoid structure.

Dowty 1979 ch 1), the introduction of another intermediate level of representation (see Fig. 1) can hardly be justified in a computational setting, and I will continue to use the simple model depicted in Fig. 2.

(1)



(2)



In fact, there is nothing to stop us from interpreting surface strings directly in the model, so even the logical form employed here needs some independent justification other than purely semantic considerations.[7] I submit that the primary motivation for an intermediate level should be the use it can have as interlingua in machine translation. In other words, the 'logical form' of natural language expressions should contain those (and only those) pieces of information which are relevant for meaning-preserving translation to other natural languages. This means that in general the logical form of a source-language expression will depend on the target language as well: in particular, the grammatical categories (gender, number, case, etc.) that have to be preserved will have to be mapped on the coarsest common refinement of the category systems of the languages in question.[8]

Therefore, we have a family of interlinguas (one for every set of natural languages), rather than one fixed logical form; and as long as the common refinement of all these languages is beyond our power to define, it is expedient to work in a formalism in which various interlinguas can be uniformly expressed. Such a formalism, here called a *metainterlingua* (MIL) can be defined as follows:

1. The primitive obs (atoms) of MIL form a finite set A. (Later on, we might equip A with some internal structure – at this point, however, the only restriction on the atoms is that they should be unique, and distinct from each other.)

2. The primitive functives of MIL are &, =, and perhaps also finitely many operations $P_1, P_2, \ldots P_n$ – these are all supposed to be binary. & will also be denoted by P_0: H is a (metalanguage) variable ranging over the $P_i (i = 0, 1 \ldots n)$.

[7] In UG, direct interpretation can be defined simply as the composition of the translation and interpretation homomorphisms.

[8] This idea is not at all original with me: for the first proposal along these lines, see Mel'cuk (1960)

3. If p and q are arbitrary obs, Hpq will be an ob, and $p = q$ is an (elementary) statement. (The only predicate of MIL is '=') $x, y, z \ldots$ will be (metalanguage) variables ranging over obs.

4. The system is defined inductively: the only obs, functives, and statements of MIL are those resulting from the iterated application of 3.

5. The axioms of MIL are:

```
x = x
                          Hx&yz = &HxyHxz
&xx = x
                          H&xyz = &HxzHyz
&xy = &yx
```

6. The rules of deduction are:

```
 x = y      x = y  y = z        x = y          x = y
 -----      ------------      ---------      ---------
 y = x         x = z          Hxz = Hyz      Hzx = Hzy
```

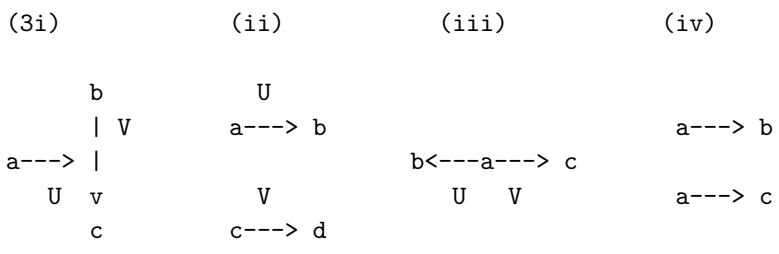
MIL, as defined above, is an equational system in the sense of Curry - Feys (1958 ch 1E): the equivalence '=' makes it possible to define a 'conjunctive normal form' with respect to &. To put it in other words, MIL is a free algebra over a finite set A generated by the binary operations $P_1, P_2, \ldots P_n$ satisfying the equations in 5: since the rules of deduction in 6 make = compatible with the operations, = is a congruence, and its classes can be represented by terms in conjunctive normal form.

To give a concrete example, if the primitive obs are taken to be the (unanalyzed) symbols PP, PA, and AA (corresponding roughly to nouns, adjectives, and adverbs), ATRANS, PTRANS, PROPEL, MOVE, ... MBUILD (corresponding to the primitive actions); and if the operations P_i are taken to be ACTOR, OBJECT, INSTRUMENT, RECIPIENT, ... POSSESSION, we get the Conceptual Dependency representation developed by Schank (1973a).[9] If we want to incorporate the theory of causality developed by Schank (1973b), we simply adjoin the operations RESCAU, ENACAU, INICAU, and REACAU.

The role of & will be illustrated on the semantic representation proposed by Kálmán - Kornai (1985): here the primitive obs are taken to be the (root) morphemes of the language in question,[10] and there are only two primitive operations, namely attribution (U), and predication (V). Complex obs of the type Uab (Vab) are depicted as edges in a graph- like structure: the vertices are labelled by a and b, and the edge running from a to b is labelled by U (V).

[9] The elaborate syntax of linkages between the conceptual categories is simply ignored here: the reader is invited to construct a system of additional axioms reflecting the restrictions on the operations.

[10] Unlike Schank's system, which is claimed to be the ultimate interlingua, this representation is (as yet) monolingual.



Edges can be the starting point or the endpoint of other edges: structures like (3i) correspond to obs like UaVbc. The operation & is interpreted as (set- theoretic) union: expressions like &UabVcd are as depicted in (3ii). The atomic vertices are taken from a finite set A. Expressions like &UabVac correspond to structures like (3iii) – since atoms have to be unique, structures like (3iv) cannot be formed. The postulates in 5. and 6. serve to make those structures that differ only in the order in which they were built up indistinguishable.

5 Interpretation

In the kind of truth- conditional approach best exemplified by Montague Grammar, the typical result is that after a long series of complex calculations we end up with a formula like

Ez(chips'(z) & eat'(John',z)) & Ew(fish'(w) & eat'(John',w))

What does *eat', fish',* or *chips'* mean here? From this analysis, all we can learn is that they are constants (of various types). But except for a few 'logical' words like *not* or *therefore* we can hardly define them in the metalanguage, and there seems to be no way to specify their extensions at every index. The usual solution is to treat such constants as abbreviations for more complex formulas: for instance in Dowty (1979 ch 4) *kill* is translated using the paraphrase

CAUSE BECOME $\neg alive'$

Here BECOME can be defined logically, because the logical system employed in Montague Grammar has some resources for handling temporal relations. CAUSE is a borderline case, but primitives like ANIMATE or HONOURED are clearly nonlogical.[11] In any case, we will have some residual constants like *alive'* – can we specify their extensions at every index?

The evidence (e.g. Labov 1973) suggests that we can not: the limits of the the extensions of words are, at best, fuzzy. Whether this fuzziness can be captured by free interpretation in models constrained by meaning postulates remains to be seen. However, it is clear from the outset that that the number of meaning postulates

[11] It appears that they do not even have a logic of their own; unlike temporal notions which, to a certain extent, reflect the objective structure of time (cf. Kamp 1979), these primitives are not constrained by an externally given structure.

necessary in such an approach is extremely great, and the task of drawing inferences in accordance with all these postulates is computationally unfeasible.

The interpretation proposed here is of a rather different sort: the idea is to capture the meanings of the constants in *programs.* It is at this point that the type-free approach pays off: programs can have other programs as their input, and if we take them to be functions (e.g. LISP functions), application is in principle unrestricted. For instance, the noun *praise* and the verb *praise* mean essentially the same thing. The difference between *gain the praise* and *praise the gain* stems not from the different meanings of *praise* and *gain* (since both appear in both constructions), but rather from the fact that they appear in different positions. There is no reason to suppose that predication is commutative: $f(g)$ and $g(f)$ can (and does) mean different things. In principle, self- application is also possible: *praise the praise* or *can the can* are well- formed expressions of English. In a type-free language, $f(f)$ can be meaningful.

The main point here is that the knowledge stored in these program 'objects' is organized linguistically: in the model outlined above, attribution and predication correspond to flow of information and flow of control, respectively. Interpretation means the activation of certain programs as prescribed by the given ob of MIL: here & corresponds to parallel execution. This kind of interpretation makes good sense in the case of imperatives, where the meaning of a command is simply the action effected in response, and in the case of interrogatives, where the meaning of a question is simply the set of possible answers the program can come up with.

If we add that programs can create other programs, the meaning of indicative sentences can be defined as the representation created during the execution of the programs corresponding to lexical items.[12] In many languages, the 'contentive' lexical items themselves are type-free, and the categorial status of the constituents is determined by various formatives (function words, affixes, word order) which, aside from their grammatical meaning, are semantically empty. In the model outlined here, the interpretation of these formatives causes no problems: the syntactic structure of the surface expressions will determine only the control structure (function- argument structure) of the programs to be executed, while the information content of the expression resides in the programs encoding the meanings of the contentive lexical items.

References

Charniak, E. 1976: A framed PAINTING: The representation of a common sense knowledge fragment. Instituto per gli studi Semantici e Cognitivi (ISSCO) Working Paper 28, Fondazione Dalle Molle, Geneve

[12] Although some recent semantic theories (called 'dynamic' in Partee 1984) fit into this perspective quite naturally, the programs I have in mind are closer to the complex frames developed by Charniak (1976) than to the simple book-keeping mechanisms proposed by Heim (1982).

Curry, H. - R. Feys 1958: Combinatory logic I. North-Holland, Amsterdam
Dowty, D. 1979: Word Meaning and Montague Grammar. D. Reidel, Dordrecht
Harris, Z. 1951: Methods in Structural Linguistics. University of Chicago Press
Hausser, R. 1982: Surface compositional grammar. ms.
Heim, I. 1982: The semantics of definite and indefinite noun phrases. Unpublished PhD dissertation, Amherst, Mass.
Jackendoff, R. 1977: X-bar Syntax. MIT Press, Cambridge, Mass.
Kamp, H. 1979: Events, instants, and and temporal reference. In: Bauerli, Egli, von Stechow (eds): Semantics from different points of view. Springer, Berlin
Keenan, E. 1981: A boolean approach to semantics. In: Groenendijk- Janssen-Stockhof (eds): Formal Methods in the Study of Language, Amsterdam Mathematical Centre Tracts, 343-380
Keenan, E. 1983: Facing the truth: some advantages of direct interpretation. Linguistics and Philosophy 6, 335-372
Labov, W. 1973: THe boundaries of words and their meanings. In: Bailey - Shuy (eds): New ways of analyzing variation in English. Georgetown University Press.
Landsbergen, J. 1977: Machine translation based on logically isomorphic Montague grammars. In: Horecky (ed): COLING 82: Proceedings of the Ninth International Conference on Computational Linguistics. 175-181
Kálmán L. - Kornai A. 1985: Pattern matching: a finite state treatment of "context-sensitive" phenomena. Paper presented at the Vienna syntax conference, May 18-19 1985
Mel'cuk, I. 1960: Grammatical meanings in interlinguas for automatic translation and the concept of grammatical meaning. Masinnyj pervod i prikladnaja lingvistika 4, 25-45. Reprinted in Rozencvejg (ed): Machine Translation and Applied Linguistics I. Athenaion, Frankfurt 95-114
Montague, R. 1970a: Universal Grammar. Reprinted in Montague 1974, 222- 246.
Montague, R. 1970b: English as a formal language. Reprinted in Montague 1974, 188-221
Montague, R. 1973: The proper treatment of quantification in ordinary English. Reprinted in Montague 1974
Montague, R. 1974: Formal Philosophy. (Thomason, ed). Yale University Press.
Parsons, T.1970: Some problems concerning the logic of grammatical modifiers. Synthese 21, 320-334
Partee, B. 1984: Nominal and temporal anaphora. Linguistics and Philosophy 7, 243-286
Schank, R. 1973a: The fourteen primitive actions and their inferences. Stanford AI Lab Memo 183, Stanford University, CA
Schank, R. 1973b: Causality and reasoning. Instituto per gli studi Semantici e Cognitivi (ISSCO) Working Paper 1, Fondazione Dalle Molle, Castagnola, Switzerland
Shubert, L. - F. Pelletier 1982: From English to Logic: Context Free Computation of 'Conventional' Logic Translation. AJCL 8, 27-44

Turner, R. 1983: Montague semantics, nominalizations and Scott's domains. Linguistics and Philosophy 6, 259-288

Williams, E. 1983: Semantic vs. syntactic categories. Linguistics and Philosophy 6, 423-466