Linguistische Datenverarbeitung

In Zusammenarbeit mit der Gesellschaft für Linguistische Datenverarbeitung (GLDV) e. V. herausgegeben von Peter Hellwig, Heidelberg und Jürgen Krause, Regensburg

Band 7

Ursula Klenk, Peter Scherber,
Manfred Thaller (Hg.)
Computerlinguistik und
philologische Datenverarbeitung

1987
Georg Olms Verlag Hildesheim · Zürich · New York
Gesellschaft für Linguistische Datenverarbeitung e. V.

Ursula Klenk, Peter Scherber,
Manfred Thaller (Hg.)

Computerlinguistik und philologische Datenverarbeitung

Beiträge der Jahrestagung der Gesellschaft für Linguistische Datenverarbeitung e. V.
1986 in Göttingen

1987
Georg Olms Verlag Hildesheim · Zürich · New York
Gesellschaft für Linguistische Datenverarbeitung e. V.

Linguistische Informationswissenschaft kommt die Arbeit von *J. Krause et al.*, in der bekannte Informationserschließungsverfahren verglichen und evaluiert werden.

Nun gutes Lesen!

Ursula Klenk
Peter Scherber
Manfred Thaller

INHALTSVERZEICHNIS

Schwarz, Christoph (1982): Freitextrecherche – Grenzen und Möglichkeiten. Anmerkungen aus der Sicht der Informationslinguistik. In: Nachrichten für Dokumentation 33: 228-236.

– (1984): Linguistische Hilfsmittel beim Information Retrieval. In: Nachrichten für Dokumentation 35: 50-51.

Schwitalla, Johannes (1979): Dialogsteuerung in Interviews. Ansätze zu einer Theorie der Dialogsteuerung mit empirischen Untersuchungen von Politiker-, Experten- und Starinterviews in Rundfunk und Fernsehen. München.

Steger, Hugo (1983a): Konzeptionelle und methodische Anforderungen an einen regionalen Sprachatlas. In: Forschungsbericht (1983): 1-15.

– (1983b): Über Textsorten und andere Textklassen. In: Textsorten und literarische Gattungen. Dokumentation des Germanistentages in Hamburg vom 1. bis 4. April 1979. Berlin: 25-67.

Steger, Hugo et al. (1974): Redekonstellationen, Redekonstellationstyp, Textexemplar, Textsorte im Rahmen eines Sprachverhaltensmodells. In: Jahrbuch des Instituts für Deutsche Sprache 1972: 39-97.

Werlen, Erika (1984): Studien zur Datenerhebung in der Dialektologie. (Zeitschrift für Dialektologie und Linguistik. Beihefte. 46.) Wiesbaden.

FINITE STATE SEMANTICS

András KORNAI
Institute of Linguistics
Hungarian Academy of Sciences, Budapest

0. Introduction

While certain theories of knowledge representation (KR) lend themselves to computer implementation quite naturally, others require a great deal of programming effort and ingenuity. Unfortunately, it is nearly impossible to evaluate or compare various KR proposals on this basis, since ease of implementation depends crucially on the architecture of the machine chosen, be it an actual computer, or a theoretical model of computation. However, there is one architecture which deserves special attention, namely that of the human brain.

The goal of this paper is to outline a theory of knowledge representation aiming at *neurological reality*: this means, among other things, that it has to be built from strictly finite state components. The calculus employed for knowledge representation is described in Section 1, and its interpretation is discussed in Section 2. The rest of this introduction outlines the neural model employed.

Neurons form a highly interconnected network in which each cell can receive impulses from certain neighboring cells and can send impulses to other nearby cells. At any given moment, a single cell, depending on its internal state and on the impulses it receives through its dendrits, will either send an impulse through its axon to the dendrits of other cells (will 'fire'), or will send no impulses at all (will 'be quiet'). This digital behaviour is captured in the classical McCullogh/Pitts (1943) model of nerve nets: neurons can be taken as finite automata having prescribed transition functions, and the nerve net itself is equivalent to a single finite automaton (cf. also *Kleene* 1956).

The process of learning can be easily modelled by rearranging the dendrits: anatomical studies, however, make it clear that after birth no such rearrangement ever takes place. How can, then, humans acquire knowledge? The answer is provided by the *kindling* model of Goddard

(1967, 1975), which puts the classical summation effect of nerve impulses in a new perspective. Subliminal impulses on a given dendrit have no effect on the output, but if we give subliminal stimuli repeatedly, then the sum of these might result in a permanent decrease of stimulus threshold. In effect, neurons can be reprogrammed, i.e. their transition functions can be changed.

To give a (hypothetical) example, suppose a neuron has two dendrits, one axon, and transition function f(0,1) = f(0,0) = 0, f(1,0) = f(1,1) = 1. This cell is insensitive to the impulse from the second dendrit, and simply transmits the impulse from the first dendrit: we might say that it stores a single bit corresponding to, say, zero. However, if we stimulate the second dendrit repeatedly, the transition function might become f'(0,0) = f'(0,1) = 1, f'(1,0) = f'(1,1) = 0. The next transition function is still insensitive to the impulse from the second dendrit, but now acts as an 'inverter' w.r.t. the impulses on the first dendrit: in effect, we changed the bit stored in the neuron from 0 to 1.

Thus, the kindling mechanism makes it possible to store a single bit permanently - this can be trivially extended to any finite sequence of binary digits. However, there is no reason to suppose that the human brain contains flip-flops, multiplication circuits, or the like. On the contrary, the functional units of the human brain, henceforth called *modules*, are expected to be highly similar to each other. These functional units will be discussed in Section 2.

With the implantation of microelectrodes, it is possible to measure whether a single cell fires or not at any given moment. In fact, it is possible to measure the activity of several hundred cells simultaneously, and with the advancement of experimental techniques, it might well become possible to measure all the relevant cells in real time. A theory can lay claim to neurological reality only if it is capable of predicting the firing pattern of all neurons involved: such predictions, at least in principle, are subject to empirical testing.

The theory outlined below cannot (as yet) make such predictions: therefore it makes no direct claim to neurological reality. The reason for this is that the critical step of circuit design from finite automata to modules will be omitted - given the lack of empirical data bearing on this matter, a detailed proposal would only have one chance in a billion to be the right one. However, since modules with the desired properties can be built from elementary (McCullogh/Pitts) finite automata , the theory does make an indirect claim 'modulo circuit design' to neurological reality, and this sets it apart from less constrained (nonfinite-state) KR theories.

1. The calculus

The logical calculi used for KR purposes 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), but so far, no type-free calculus has been employed for this purpose. The aim of this section is to outline a transfer language (called MIL) for machine translation which is based on combinatory logic. The main advantage of the type-free approach is that the traditional distinction between rules (e.g. phrase structure rules) and representations (e.g. that of lexical items) disappears: every chunk of knowledge relevant to natural language understanding and related tasks is stored in the same format, namely as an ob of the system. Before turning to the details of MIL, however, it will be necessary to discuss some of the problems with the traditional (typed) approach, as exemplified by Montague Grammar.

The type theory of Montague (1970a, henceforth UG) is based on the assumption that the world is a collection of 'things': these will correspond to the individual constants of the model. Since we can give a name to every thing, *proper names* will have type e. *Common names* are usually taken to represent collections of things: therefore, they are interpreted as sets of individuals and will have the type (e → t). It is less clear what to do with abstract nouns like *love*, and the decision here is usually based on the linguistic similarity of abstract and concrete nouns, so that the former are also taken to be of type (e → t). This is in sharp contrast to the frame-semantic approach embodying the common sense observation that love is between individuals, for that would give us a relation of type eXe.

Adjectives combine with nouns into phrases syntactically equivalent to nouns: using the notations of categorial grammar, A = N/N. Since they take (e → t) elements as their arguments, and their values must also be of type (e → t), adjectives will have the type (e → t) → (e → t). [1]

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 → u). I will call this requirement the *soundness* of type-assignment. But this requirement is insufficient for unique type-assignment unless we stipulate that the only permitted mode of composition is *application* [2] (of a function to an argument), and we

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

[2] 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).

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 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.

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 practically every version of the theory. 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 four, or even more slashes is quite common.

In fact, *Turner* (1983), using the process of nominalization in English shows that if we map English expressions into formulas interpreted in Russellian typed models, than certain natural requirements on this mapping can not be simutaneously met. (For a fuller discussion, and a similar demonstration based on category changing suffixes in Hungarian, see *Kornai* 1985). To solve this problem, *Keenan* (1981, 1983) enriched Russellian type theory considerably, and *Turner* (1983) employs Scott domains as models. My proposal is to do away with types altogether: this can be achieved by adopting a type-free model.

For the purposes of machine translation, 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. [3] 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,

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

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 functions of MIL are &, =, and perhaps also finitely many operations P1, P2, ..., Pn - these are all supposed to be binary. & will also be denoted by PO. H is a (metalanguage) variable ranging over the Pi (i = 0,1,...,n).

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, ... 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 = &yz

6. The rules of deduction are:

$$\frac{x = y}{y = x} \qquad \frac{x = y \;\; y = z}{x = z} \qquad \frac{x = y}{Hxz + Hyz} \qquad \frac{x = y}{Hzx = Hzy}$$

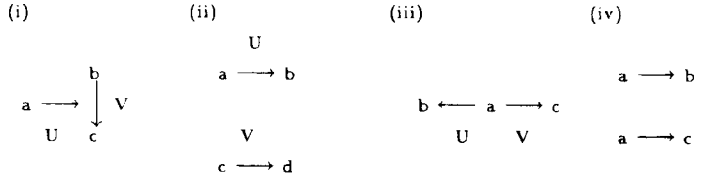
MIL, as defined above, is an equational system in the sense of *Curry / Feys* (1958 ch IE): the equivalence '=' makes it possible to define a 'conjunctive normal form' with respect to &. To put it in other words, MIL is a free algebra over a finite set A generated by the binary operations P0, P1, ..., Pn 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 Pi are taken to be ACTOR, OBJECT, INSTRUMENT, RECIPIENT, ..., POSSESSION, we get the Conceptual Dependency Representation

developed by *Schank* (1973a). [4] 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 employed by *Kálmán/Kornai* (1985): here the primitive obs are taken to be the (root) morphemes of the language in question, [5] 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 called a *dependency graph*. The vertices are labelled by a and b, and the edge running from a to b is labelled by U (V).

(1)



Edges can be the starting point or the endpoint of other edges: structures like (i) correspond to obs like UaVbc. The operation & is interpreted as (graph-theoretic) union: expressions like &UabVcd are as depicted in (ii). The atomic vertices are taken from a finite set A. Expressions like &UabVac correspond to structures like (iii) – since atoms have to be unique, structures like (iv) 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.

To give an example, suppose we learn that horses drink only clear water. How do we represent this knowledge? 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

(horse'(z) & water'(w) & drink'(z,w)) ⇒ clear'(w)

But what does *horse'*, *water'* or *clear'* 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

[4] 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.

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

metalanguage, and it is doubtful whether we can specify their extensions at every index. In fact, the evidence (e.g. *Labov* 1973) suggests that we can not: the limits of the extensions of words are, at least, 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 the number of meaning postulates necessary in such an approach is extremely great, and that the task of drawing inferences in accordance with all these postulates is computationally unfeasible.

In the dependency graph notation outlined above, we have

(2) horse → drink ← water ← clear

(The 'only' part can be represented as

(3) horse → drink ← water

but I will not discuss this matter here.)

The interpretation of dependency graphs proposed here differs radically from model theoretic interpretation: the idea is to capture the meanings of the constants in *programs*. Notice, that 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. Only in a type-free language can f(f) be meaningful.

2. The interpretation

The minimal anatomical units of the brain are its cells – the minimal functional units (called modules), however, are expected to be much larger (ca. 10 cells/module). What are the general properties of these modules? In what follows, I will suppose that ordinary lexical entries correspond to single modules, and will derive the basic characteristics of modules from this assumption. Lexical entries contain at least phonological and semantic information: the former can be used e.g. to control the articulators, and the latter e.g. to draw inferences. While the phonological representation of lexical entries is narrowly constrained (for a specific

proposal, see *Halle* 1983), the representation of meaning, as far as we know, is not. In certain instances (e.g. *hand*) it might be connected to the body scheme, in others (e.g. *red*) to the visual perception system, in yet others (e.g. *tuesday*) only to similar highly abstract entities. Since we are concerned mainly with semantics here, it seems unreasonable to constrain the semantic representations in advance: for the time being, I will suppose that the meaning of lexical items – stored as their special information, or *spinfo* – can be any partially recursive function, i.e. any LISP program. Naturally, there are length restrictions on these programs, but as these should be stated in terms of the neural machine code, here it is sufficient to say that the spinfo in some given module can not exceed a finite uniform bound.

Programming objects [6] can communicate with each other quite freely – naturally, this is not the case with real-life modules. Among these, communication is strictly local: each one can send (or receive) messages only to (from) some of its neighbors, which are listed on the *neighbors list* once and for all. At this point, it might be useful to draw an ana logy between modules and the processors of massively parallel computers such as the Connection Machine: each processor (module) has a limited storage area for itself (spinfo), and is connected to a limited number of other processors in hardware (neighbors).

The message a modul can send has to be built from its current spinfo: in particular, the internal states of other modules can have no direct effect on it. In an idealized model, where the modules are synchronized, everything happens in two cycles: in odd cycles, the modules digest the information received from neighbors, and in even cycles, the modules send and receive messages. This means that the internal states of module M1 at T(1) can affect the workings of module Mn (to which M1 is only indirectly linked via intermediate modules M2, M3, ..., Mn-1) only at time T(2n-1): there is no such thing as instantenous long-distance communication in this model.

In addition to its spinfo, each module is characterized by its current *activation state*, which is encoded by a set of *flags*. For instance, when someone hears the sentence

(4a) John hates porcupines

the state of the module corresponding to *porcupine* is somewhat altered.

[6] The system is being implemented in an object-oriented programming environment, the so-called Beggarman's Flavor System, which is based on the Poor Man's Flavor System (*Di Primio/Christaller* 1983), and runs on the IBM 3031 of the Computer Science Institute of the Hungarian Academy of Sciences.

This can be demonstrated by purely linguistic methods: if we continue with

(4b) but I think they are lovely

they refers to porcupines. and not to, say, rats. This difference must be reflected somewhere in the system: either we say that the flag TOPIC is set differently on *rat* and *porcupine*, or we say that a copy of the *porcupine* spinfo now resides in some other module (and this is not true for the spinfo of *rat*). If we choose the latter option (see e.g. *Kálmán* 1985), there is no need for a TOPIC flag – however, the TEMPORARY flag has to be set on the programmable module in which the spinfo of *porcupine* temporarily resides. The name of these flags is immaterial: in any case, there can be only a few flags which, between themselves, can only distinguish a very limited number (< 10) of internal states. The most important of these states is the passive or *basic* state (denoted by F0): unless a message received by a passive module contains some specific reference to the contents of this module, the message will not be forwarded and the module remains in state F0. Other states of interest are the stimulated (F1), inhibited (F2), and learning (F3) states. In a more realistic asynchronous model, flags can be used to signal whether a given module is currently in the processing stage, or whether it is capable of receiving messages, etc.

The central item in the LISP program giving the spinfo is a list (called lista) of associated modules: in fact, most modules that have no efferent connections will only have a lista in their spinfo. That means that if we do not want to describe how one raises his hand upon hearing the command *Raise your hand*, i.e. if we restrict ourselves to matters of 'pure' semantics, then only the association lists are available to us. Therefore, we have a highly restricted knowledge representation in which everything has to be achieved by manipulating listas. Moreover, as these lists are in principle unordered, [7] this theory is based on a finite set M of primitives (corresponding to Modules or Morphemes). M is equipped with a binary relation N (neighbors), and an association relation A which is an extension of N. Modules can send messages only to their immediate neighbors: if we want a message to arrive at some unlinked module, we have to address it properly.

To continue with the above example, presumably, neither *water* nor *drink* was associated to *horse* previously. We can suppose, however, that *drink* is associated to the abstract modules SUBJECT and OBJECT – in fact, such a statement must be part of the lexical entry of *drink*. For

[7] Naturally, they are implemented as ordinary lists on the computer.

our purposes, it does not really matter whether the association between drink and e.g. OBJECT is direct (i.e. they are neighbors) or is effected by intermediary modules (such as VERBINTR) – the point here is that OBJECT appears on the lista of *drink*. OBJECT and SUBJECT are, in turn, associated to NOUN. Although NOUN appears on the lista of both *horse* and *water*, there is no reason to suppose that either *water* or *horse* appears on the lista of NOUN – clearly, that would make the latter lista much too long. Similarly, the only link between *water* and *clear* is provided by the chain

(5) water → NOUN ↔ MODIFY ↔ ADJECTIVE ← clear

– as can be seen, this gives us no direct path from one to the other. But, in the course of parsing (for details, see *Kálmán/ Kornai* 1985), the contents of both *water* and *clear* will appear in the module MODIFY, which will send it to NP, and so on. In the end, the whole parse will arrive to *horse* via the module SUBJECT. 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. The main point here is that the knowledge stored in these program 'objects' is organized linguistically: in particular, it is stored in the lexical entry corresponding to the subject [8] of the sentence. The list-structure of the parse corresponds, to a surprisingly large extent, to the syntactic constituent structure of the original sentence, and the *meaning* of sentences can be defined as the representation created during the execution of the programs corresponding to lexical items.

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 presented 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

Curry, H. / Feys, R. (1958): Combinatory logic I. North-Holland, Amsterdam.

[8] Or rather, the communicative theme, or 'topic' (see *Kálmán* 1985).

Di Primio F. / Christaller Th. (1983): A poor man's flavor system. ISSCO Working paper 47 Institute Dalle Molle, Geneva.

Goddard, V. G. (1967): Development of epileptic seizures through brain stimulation at low intensity. In: Nature 214: 1020-1021.

Goddard, V.G. / Douglas, R. N. (1975): Does the engram of kindling model the engram of long term memory? In: Kindling. A Symposium on Basic Research in Neuroscience. Canadian Journal of Neurological Science 2: 383-394.

Halle, M. (1983): Distinctive features and their articulatory implementation. In: Natural Language and Linguistic Theory 1: 91-107.

Hausser, R. (1982): Surface compositional grammar. Ms.

Kálmán, L. (1985): Type free context change semantics. Ms, HAS Institute of Linguistics.

Kálmán, L. / Kornai, A. (1985): Pattern matching: a finite state approach to parsing and generation. Ms, HAS Institute of Linguistics.

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. In: Linguistics and Philosophy 6: 335-372.

Kleene, S.C. (1956): Representation of events in nerve nets and finite automata. In: *Shannon / McCarthy* (eds): Automata Studies, 3-41. Princeton University Press.

Kornai, A. (1985): Logical types and linguistic types. To appear in *I. Rusza* (ed): Tertium Non Datur.

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. (1982): Machine translation based on logically isomorphic Montague grammars. In: *Horecky* (ed): COLING 82: Proceedings of the Ninth International Conference on Computational Linguistics, 175-181.

McCullogh, W.S. / Pitts, W. (1943): A logical calculus of the ideas immanent in nervous activity. In: Bulletin of mathematical biophysics 5: 115-133.

Mel'cuk, I. (1960): Grammatical meanings in interlinguas for automatic translation and the concept of grammatical meaning. Mašinnyj perevod i prikladnaja lingvistika 4: 25-45. Reprinted in *Rozenzvejg* (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.

Partee B. (1977): John is easy to please. In: *A. Zampolli* (ed):. Linguistic structure processing. North-Holland, Amsterdam, 281-312.

Schank, R. (1973a): The fourteen primitive actions and their inferences. Stanford AI Lab Memo 183, Stanford University (CA).

Schank, R. (1973b): Causality and reasoning. ISSCO Working Paper 1, Fondazione Dalle Molle, Castagnola (Switzerland).

Schubert, L. / Pelletier, F. (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. In: Linguistics and Philosophy 6: 259-288.

WAS LEISTEN INFORMATIONSLINGUISTISCHE KOMPONENTEN BEI DER INHALTSERSCHLIESSUNG FÜR EIN DEUTSCHES PATENTINFORMATIONSSYSTEM?

J. KRAUSE, C. SCHNEIDER, G. SPETTEL, C. WOMSER-HACKER
Universität Regensburg

1. Überblick und Problemstellung

Obwohl informationslinguistische Komponenten bei der Inhaltserschliessung von Massendaten eine lange Tradition haben, ist bis heute unklar, ob es sich lohnt, die verbreiteten Freitextverfahren mit ihren ergänzenden Retrievalfunktionen wie Trunkierung und Kontextoperatoren durch linguistisch begründete Teilkomponenten (z.B. die Reduktion von Wortformen auf ihre Grundformen oder eine Nominalgruppenanalyse) zu ersetzen bzw. zu ergänzen. Die Meinungsvielfalt reicht von krasser Ablehnung bis zum naiven (nicht hinterfragten) Glauben an die These, ein Mehr an informationslinguistischen Komponenten müsse zwangsläufig zu besseren Retrievalergebnissen führen. Dem Mangel an Konsens unter den Wissenschaftlern entspricht ein Mangel an empirischen Daten zur Beantwortung dieser Frage.

Deshalb konnte auch die Entscheidung, welches Inhaltserschließungsverfahren beim Aufbau eines Deutschen Patentinformationssystems (DPI) die günstigsten Retrievalergebnisse verspricht, nicht ohne extensive Tests der derzeit zur Verfügung stehenden Systeme beantwortet werden. Diese Ausgangssituation führte zum Projekt PADOK, das vom 1.1.1985 - 31.3. 1986 am FG Linguistische Informationswissenschaft der Universität Regensburg durchgeführt wurde (gefördert unter BMFT Nr. 10 131 14). Durch PADOK sollte eine wissenschaftlich fundierte Entscheidungsgrundlage für das Bundesforschungsministerium und das Patentkonsortium, das den Aufbau eines DPI betreibt, geschaffen werden. Zugleich ergaben sich empirische Erkenntnisse über die Wirksamkeit informationslinguistischer Teilalgorithmen zur Inhaltserschließung von Massendaten.