

Computational Linguistics Computerlinguistik

An International Handbook on Computer Oriented
Language Research and Applications

Ein internationales Handbuch zur computergestützten
Sprachforschung und ihrer Anwendungen

Edited by / Herausgegeben von
István S. Bátori · Winfried Lenders
Wolfgang Putschke

Offprint/Sonderdruck

Walter de Gruyter · Berlin · New York
1989

VI. Computer-Aided Description of Language Systems III: Testing of Rule Systems Computergestützte Beschreibung von Sprache III: Testen von Regelsystemen

21. Computational Testing of Linguistic Models in Morphology

1. Introduction
2. Kay's Morphosyntactic Chart Parser
3. Kaplan and Kay's Morphographic Transducer
4. Koskenniemi's Two-Level Model
5. Hellberg's Paradigm Model
6. Literature (selected)

1. Introduction

The basic task facing any computationally implemented morphological model is that of analyzing the word forms of the individual language(s). The process of word form analysis is often broken down into subtasks: segmentation of morphs and morphological alternants of various types, identification of stems and endings by dictionary look-up (lemmatization), consistency checks between the morphological elements established, and retrieval of lexical information. In an optimal morphological model, consideration should also be paid to idiom recognition and other matters interfacing morphology with syntax and semantics.

In the early days of computational morphology, most work was done in fairly language- and machine-dependent (occasionally even brute-force) ways. Not much attention was paid to isolating the linguistic core of the models from programming details. Typical features of many approaches were that only restricted subparts of the vocabulary were covered, and that the complexity of morphological processing was underestimated (especially so if the language under analysis was of the English type with negligible inflectional morphology). A fairly common view has been what could be called the full listing hypothesis (FLH) according to which word forms are just listed in the lexicon. Under this view, no genuinely morphological processing would be needed (cf. Winograd (1983,

544 ff.) for a recent statement to this effect). However, the predominant view is that some morphological processing is needed even in parsers for languages like English.

The classical taxonomy of morphological models is the tripartition into word-and-paradigm (WP), item-and-arrangement (IA), and item-and-process (IP). Variants of IP have been the most popular ones in computational morphology (discussed in 2.—4.). WP has also been entertained in a full-blown form (in 5.). However, this is not intended to imply that computational morphology would be equal to a straightforward take-over and implementation of autonomous morphological theories. Computational morphologists have designed a partly novel body of concepts and theoretically relevant tools especially in the domain of lexical representation and dictionary look-up. Very little work has, in fact, been done on direct computational testing of autonomously established linguistic models. We shall therefore assume a somewhat broader perspective and treat the topic from a more independent computational view. — Computational morphology has been strongly inclined towards analysis. The problems of morphological synthesis (production) will not be dealt with here except in passing (but cf. Karttunen/Root/Uszkoreit 1981; Koskenniemi 1983 a; 1985 b). Likewise, we shall treat derivation and compounding only superficially (but cf. Rüdiger 1975; Hellberg 1978; Hoepfner 1980; 1982 a; Reimann/Rüdiger 1982; Sparck-Jones 1983).

2. Kay's Morphosyntactic Chart Parser

One of the outstanding achievements of computational morphology and syntax so far is

Martin Kay's chart parser which provides a general integrated morphosyntactic parsing framework. Most of the central concepts of computational morphology originate in the work of Kay and his associates (Kay/Martins 1970; Kay 1973; 1977; Kaplan/Kay 1981).

The chart is a data structure (a directed graph) consisting of vertices and labelled edges (arcs). It provides a 'board' where all results, morphological as well as syntactic, of the ongoing analysis are put. In ambiguous situations all interpretations are recorded in the graph. There is no backtracking nor any recomputing of structures once analyzed.

A typical feature of Kay's approach is his use of morphographemic rules similar to those of standard generative phonology. This justifies classifying chart parsing as IP from the morphological point of view. In a sense, chart morphology could be viewed as computational (implementation and) testing of some of the basic tenets of generative phonology. — E.g. English would thus be morphologically parsed by postulating morphographemic rules for alternations such as *funny* : *funni + ly*, *try* : *trie + s*, *fus + es* : *show + s*, *labe l* : *labell + ed*. Application of these (and other) rules under proper circumstances guides the introduction of edges into the chart. E.g., given a rule with the effect *ied ≠ ---> y + ed ≠* (where \neq denotes word-final position), the chart might look as follows, in the relevant respects, upon analyzing the word *tried* (Kay 1977, 141):

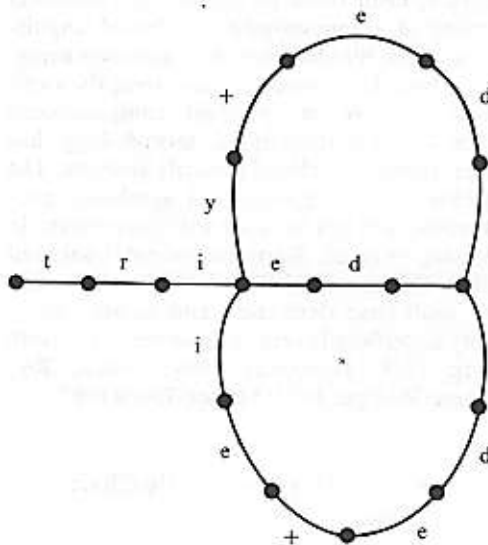


Fig. 21.1: Schematic state of the chart after morphological rewriting (Kay 1977, 141)

An edge is established once it is found out that there is a rewriting rule applicable in a proper context. The new edge is labelled with the material to the right of the rewriting arrow. The whole process is driven by executing tasks in a stack-structured agenda.

Kay has devoted considerable effort to developing and optimizing the process of dictionary lookup. When large dictionaries are used (containing tens or even hundreds of thousands of lexemes), it is of course essential that dictionary lookup does not become impracticable. The method now standardly used is to construct a binary letter tree where shared initial substrings of lexical entries are conflated. A letter tree for the lexicon /a aardvark aback abacus abaft ace acetic acid acidity / would look like this (Kay 1977, 162):

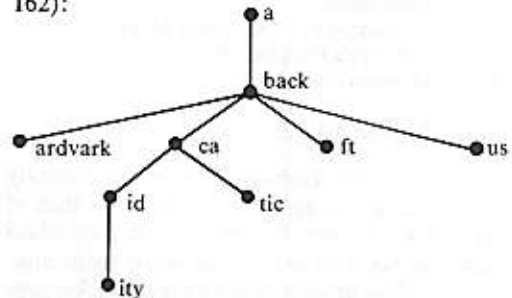


Fig. 21.2: Letter tree with conflated shared initial parts for a small lexicon (Kay 1977, 162)

Many of these techniques are part and parcel of ordinary computational search methodology. However, they have important ramifications also for morphological theory and for explicit theories of language recognition. It clearly cannot be the case e.g. that the hearer/reader is just scanning sequential lists resembling ordinary dictionary books when he/she is decoding linguistic messages (cf. 4. below).

A third important feature is the idea of employing linked minilexicons for expressing morphotactic restrictions and ruling out syntagmatically improper interpretations in instances of ambiguity (e.g., English *like* may be a preposition, verb, adjective, or noun, but only the verbal interpretation is proper in the word form *liking*). The earliest explicit large-scale exposition of this idea may be found in Karttunen/Root/Uszkoreit's (1981) TEX-FIN program which is a complete morphological analyzer and synthesizer for Finnish (but also cf. e.g. Kay 1977, 168).

Finally, the chart parsing formalism is notable for its explicitness in the morphology-syntax interface. Morphological anal-

ysis is only one part of the parsing process. Its output has no special value outside the global context of syntactic structuring and semantic interpretation. Already the Kay/Martins' (1970) paper provides a detailed discussion of matters such as consistency checks of adjacent morphological elements, suppression of irrelevant ambiguities, retrieval of morphological information from the lexicon, and proper design of it for further syntactic and semantic processing. These are areas where autonomous grammar models tend to have little to offer but which are highly relevant once one broaches the task of integrating all the subsystems of language structure. In fact, we would see one of the main (both theoretical and practical) benefits of computational modelling in the possibility of interfacing all subsystems and making them work as a unified whole on sufficiently large domains (in terms e.g. of lexicon size and of coverage of a vast number of full and reduced clause structures).

3. Kaplan and Kay's Morphographemic Transducer

Kaplan/Kay's (1981) paper on phonological rules as finite state transducers has had a major impact on computational morphology in the 1980s (cf. Kay 1983b). Theoretically, its significance lies in the possibility of translating ordinary phonological rules (in the generative sense, i.e. morphophonological rules) into two-tape finite-state transducers similar to finite-state machines except for the number of tapes. This provides an independent way of construing (one aspect of) the notion 'morphological complexity'. In particular, it conveys the interpretation that morphophonological alternations are "simple" in the sense that (many of them) may be modelled by formally very simple abstract machines. Fig. 21.3 displays Kay's (1983b, 100) transducer for (a simplified version of) the rules governing the spelling of the endings of regular English plural nouns and third person singular verbs. The designated characters S and Y have obvious morphophonological interpretations.

The circles denote states, double circles potential final states. Character pairs refer to corresponding characters on the lexical level (represented as a search tree, cf. 2. above) and the level of surface graphemes. Arcs show state transitions when a certain pair (lexical-surface character correspondence) is en-

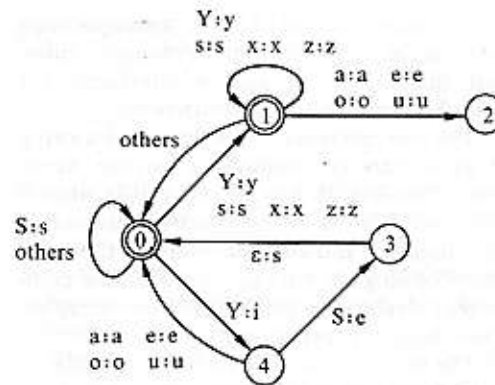


Fig. 21.3: Transducer for some English morphographemic rules (Kay 1983b, 100)

countered. Epsilon denotes the empty symbol (i.e. 'nothing'). The label "others" refers to lower-case character pairs not mentioned on other arcs. Supposing that the lexical representation of the verb *fly* is *fly*, and that the transducer is fed with the surface *flies*, one sees that the character pairs *f:f* and *l:l* trigger no state transitions from the initial state 0. There is an allowed correspondence of lexical *Y* and surface *i* in state 0 with a subsequent transition to state 4. From this state there is i.a. a transition back to 0 if the current character pair *e:e*. Finally, the pair *s:s* in state 0 does not change the situation. Now the input string has been consumed and the system is in one of the designated final states. Therefore the transducer accepted the word form *flies* (furthermore, relevant routines of course retrieved a proper morphological representation from the dictionary).

One of the interesting ideas of this approach is that the lexicon is not just a passive depository of items. Rather, it directs the operation of the transducer by actively evaluating the feasibility of encountered lexical-surface character pairs.

At least theoretically, there is a possibility of cascading all the different transducers needed for the morphological analysis of a language into a single large automaton where the number of states is in the hundreds or perhaps even the thousands.

4. Koskenniemi's Two-Level Model

Kimmo Koskenniemi's (1983a, b) two-level model for morphological analysis and synthesis is in some respects an elaboration of themes treated in 2.—3. above. In particular,

this concerns the use of finite-state automata for simulating morphophonological rules, and the use of linked minilexicons for describing morphotactic restrictions.

The two levels are the lexicon and a string of characters constituting a surface word-form. The lexicon has an active role since it also specifies the morphotactic structure of the language and includes part of the morphophonological alternations. The rule component deals with the bulk of the morphophonological alternations.

The two-level model is a unified model of word-form recognition and production. That is, the same linguistic description may be concretely run both as a word-form recognizer and a producer. This is a major novel feature. In modelling language use, it is essential to strive for some level of description where recognition and production meet because it would be clearly unrealistic to claim that recognition and production are distinct to the extent of invoking separate lexicons.

A central feature of the two-level model is that it provides a general, language-independent formalism for morpholexical description. The model supplies a computer program which can utilize descriptions of individual languages made according to a certain linguistically motivated formalism. The formalism provides a format for lexical entries and their possible continuation patterns, and rules for expressing correspondences between the lexical and surface levels.

The task of the lexicon system in the two-level model is more extensive than in most other models (except for Karttunen/Root/Uszkoreit's (1981) TEFIN model). In addition to listing lexical entries, it defines the morphotactic structure of word-forms as a network of pointers. The lexicon is thus an active lexico-morphological component. The two-level rule component treats fairly natural, transparent alternations, but in a manner different from that of generative theory. The rules do not perform sequential operations on strings but rather express permissible correspondences between the two levels. Nonnatural, opaque stem alternations are treated in the lexicon by listing the alternatives as such.

Given a lexicon L with any number of entries, the two-level model purports not only to recognize and produce all the grammatical word-forms in relation to L, but also to spot all ungrammatical ones. 'Ungrammatical' may mean either running counter to

the specifications given by L and the rules, or lacking an entry in L.

The current scope of the two-level model is restricted to phonemic (or graphemic) surface representations and is based on the following simplified classification of morphophonological and morphological alternations:

(i) 'Fairly natural' one-segment modifications: mostly automatic, transparent, productive, exceptionless alternations between phonologically closely related single phonemes in predominantly phonological contexts.

(ii) Suppletion-like unnatural alternations: alternations between either longer sequences of phonemes or between phonologically unrelated (or distantly related) single segments. These alternations are often opaque and unproductive.

Only the first type is described by rules in the two-level model. Suppletion-like alternations are stated as lexical patterns equivalent to stem sets. Of course, there are borderline cases.

Alternations of type (ii) may occur in quite common and productive inflexional types. E.g. all Finnish nouns ending in *-nen* are subject to a suppletion-like *nen-se* alternation. These words are both common (thousands of entries) and productive (e.g. in derivational suffixes such as *-lainen* 'inhabitant of', *-mainen* 'like'). The relevant stems are suppletively related:

hevonen nominative singular 'horse'
hevose + n genitive singular 'of a horse'

No natural rules relate medial *n* and *s*, or delete word-final *n*, in such environments. To avoid ad hoc rules for transforming sequences of phonemes into other phonemes, the two-level model uses ALTERNATION PATTERNS. An alternation pattern is essentially a minilexicon containing the lexical representations of the alternatives. An alternation pattern is usually common to all members of a certain inflexional type. The root entries of the nouns *hevonen* 'horse', *ihminen* 'human being', and *työläinen* 'worker' contain only the constant part of the nominative singular form (these are 'technical stems' in the sense of Hellberg 1978). The variable part at the end is truncated. Note that the two-level lexicon system needs only one instance of each alternation pattern, e.g. (4). All nouns of the same inflexional type refer to the same pattern. This mechanism captures a generalization just as rules or operations do, but con-

ceptually and operationally it is a part of the lexicon. No actions are implied, only references by way of pointers. The lexical continuation patterns build up networks of stems and endings of appropriate classes. Notice how this correlates with bottom-up, linear, left-to-right processing, which should be basic attributes of models aspiring to treat the real process of word-form recognition.

Inflexional endings are stored as MINILEXICONS and referred to by the lexical entries. Entries for words with only natural alternations refer to the whole set of relevant endings, e.g. for the noun *lasi* 'glass':

```

lasi---> 0 nominative singular
           n
           A oblique singular
           ssA endings
           ...
           t
           en
           IA plural endings
           IssA
           ...

```

Just as root entries may refer to ending minilexicons, the entries for endings may refer to further minilexicons. This is the mechanism used for describing morphotactic structure within the lexicon system. The reference mechanism combined with the alternation patterns proposes a solution to the problem of how to represent inflected forms in the lexicon.

In a sense, all inflected forms 'are' in the two-level lexicon because they can be retrieved and produced by following the pointers. On the other hand, there is only one entry per lexeme, and the alternation patterns and endings are shared by other entries.

Natural (or near-natural) (morpho)phonological variations are dealt with by the rule component implemented as finite-state automata. Even here there is no effective step by step processing involved, and no prediction is made that forms with morphophonological alternations should be heavier to process. The two-level model is thus intermediate between the FLH referred to in 1., and the generative Derivational Hypothesis (that derives many word forms by applying directional rules).

For the purposes of linguistic description, the lexicons are given as lists of entries. The sublexicon containing Finnish word roots is in this format:

LEXICON Root		
<i>lastu</i>	/S	'chip N';
<i>lasi</i>	/S	'glass N';
<i>leivo</i>	<i>nen/S</i>	'lark N'
...		

Some lexemes are represented simply by their nominative stem in ordinary segmental form, others by the invariant part of the stem (e.g. *hevo*) followed by a pointer to an alternation pattern (e.g. *nen/S*) which, in turn, is a minilexicon.

The two-level model postulates a lexical level defined by entries and pointers, and a surface level defined by phonemic or graphemic characters. These two levels are directly compared by the rules. Thus, no intermediate levels or stages exist.

Rules check for permitted or required discrepancies between the two levels and they operate in parallel. Each rule states a certain constraint for the correspondence between the two levels. A certain lexical representation has a certain surface form as its realization, and vice versa, when all pertinent correspondences stated by the rules are satisfied. Thus, the basic role of the rules is not to transform one representation into another, but to dictate in what relation the representations must be.

The special checking nature of the TWO-LEVEL RULES means that the rule component as such is nondirectional, but it can be used bidirectionally, i.e. both for word form analysis and generation.

The lexical representations consist of phonemes, morphophonemes, and morphological features. Features are used for encoding grammatical information such as junctures or the presence of certain classes of endings. Morphophonemes represent one-segment alternations which need lexical marking. Features correspond to surface zero, morphophonemes to ordinary phonemes.

Consider the following example. The Finnish noun *lasi* 'glass' belongs to the most common and productive type of nouns ending in *i*. The lexical representation of the partitive plural consists of the stem *lasi*, the plural morpheme *I*, and the partitive ending *A*. In the two-level framework the lexical representation *lasiIA* is written above the surface from *laseja*:

```

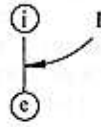
Lexical representation: lasiIA
Surface representation: laseja

```

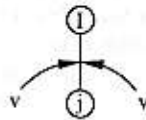
This configuration exhibits three morpho-

phonological alternations, all considered to be of the nonsuppletive type:

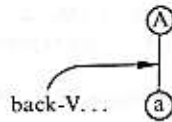
- (1) Stem final *i* is realized as *e* before the oblique plural morphophoneme *I*, schematically:



- (2) The plural morphophoneme *I* is realized as *j* between surface vowels, schematically:



- (3) The partitive (like other endings) is subject to vowel harmony. Here, a lexical archiphoneme *A* is postulated which is realized as *ä* or *a* according to stem harmony, schematically:



Two-level rules may refer both to lexical and surface representations. (1) refers to a certain lexical form, the oblique plural, not to any *i*. (2, 3) only refer to the surface. Alternations may be linked to each other, e.g. certain stem-final *e*, *a*, *ä*-vowels are realized as zero in front of the plural *I*, whereby the latter is realized as *i*.

For linguistic purposes Koskenniemi uses a special two-level rule formalism for writing morphological descriptions. The rules corresponding to (1–3) are expressed as:

$$(4) \quad \begin{array}{c} i \\ e \end{array} \quad \langle = \rangle \quad \text{---} \quad \begin{array}{c} I \\ = \end{array}$$

$$(5) \quad \begin{array}{c} I \\ j \end{array} \quad \langle = \rangle \quad \begin{array}{c} \bar{v} \\ \bar{v} \end{array} \quad \text{---} \quad \begin{array}{c} \bar{v} \\ \bar{v} \end{array}$$

$$(6) \quad \begin{array}{c} A \\ a \end{array} \quad = \rangle \quad \begin{array}{c} \bar{v} \\ \bar{v} \end{array} b \quad \left(\begin{array}{c} \bar{v} \\ -Vf \end{array} \right)^* \quad \text{---}$$

Rule (4) is spelled out as “the lexical phoneme *i* is realized as a surface *e* if and

only if it occurs in front of the plural *I*”. Rule (5) is read similarly: “the morphophoneme *I* is realized as *j* if and only if it occurs between surface vowels. Rule (6) states that “if we have a correspondence $\begin{array}{c} A \\ a \end{array}$ then there must be a preceding back vowel with no intervening front vowel” (but there may be other realizations for *A* in this environment).

There are three types of two-level rules corresponding to the three operators (i) “= \rangle ”, (ii) “<= \rangle ” and (iii) “<=>”. Rule type (i) with “= \rangle ” states that for an occurrence of the left hand correspondence pair to be acceptable, it must occur in the context given by the right hand side of the rule. Rule type (ii) with the operator “<= \rangle ” states that a correspondence pair with the lexical segment specified in the left hand side of the rule occurring in the context given in the right hand side must have a surface realization specified in the left hand side. The last type of rules (iii) with the operator “<=>” is most common. These rules combine the effects of types (i,ii), thus defining the necessary and sufficient environments for correspondences.

Koskenniemi’s two-level rule component as a whole is less powerful than standard generative phonology due to the lack of rule ordering restrictions.

The two-level model has been applied to several languages including English (Karttunen/Wittenburg 1983), Finnish (Koskenniemi 1983a), French (Östling 1985), Japanese (Alam 1983), Old Church Slavonic (Lindstedt 1985), Polish (Borin 1985), Rumanian (Khan 1983), and Swedish (Blåberg 1985).

5. Hellberg’s Paradigm Model

Staffan Hellberg’s (1972, 1978) paradigm model designed for morphological analysis of Swedish word forms is an early comprehensive approach deserving special interest in this connection since it implements a version of word-and-paradigm morphology. It is relevant also because Hellberg aims at full coverage of the morphological system, i.e. including all types of variations, limitations, and extensions. The full description contains 235 paradigms, the lexicon some 8,600 lemmas (in 1978). Almost no rewriting rules in the standard sense are used. A typical paradigm in Hellberg’s procedural format looks like this (the words to the right just exemplify the forms of the paradigm):

101 *flick/a* nn utr
 LO ≠ ---> 92 *flicka(s)*
 ---> e ---> 96 *flickan(s)*
 flickor(s)
 > 5 *flickorna(s)*
 flick(s)-
 SH ---> 11 *flicke-*
 ---> (10)

First comes a mnemonic handle for the paradigm (101). The slant in *flick/a* marks the end of the technical stem. This part of the word is common to all paradigm members and represents the lemma in the dictionary. Then proper morphological information is given (here, nn = noun, utr = non-neuter gender). The rest of the paradigm description contains references to (numbered) subroutines invoked depending upon the segmental nature and length of the tail (i.e. the rightwards context). The notion 'subroutine' is a partial functional predecessor to the minilexicons already treated in sections 2.—4. above. An expression such as "> 5" indicates the maximal segmental length of the tail for it to qualify as an inflectional coda. If the tail (in this example) is longer than 5, it is "LO(ng)" and referred to subroutines checking whether it is a derivative or a compound. Otherwise, the tail is "SH(ort)" and tested for inflectional status. These tests are performed in an order derived (roughly) from studies of textual frequency. The gable ≠ in the paradigm refers to the end of the linguistically motivatable stem. The gable has no direct function in the paradigms or subroutines aside from stressing that technical and linguistic stems are distinct.

Suppose the word token at hand is *flickornas*. The technical stem *flick* is retrieved from the lexicon. Since tail length does not exceed 5, the paradigm description calls subroutine 11 which looks like this:

11
 ---> a ---> ≠ 91 indef sing
 ≠ n ---> 91 def sing
 ---> ≠ o ---> 10

This routine accepts the segment *o* and gives a further reference to subroutine 10:

10
 ---> r ---> 91 indef pl
 ---> n ---> a ---> 91 def pl

The notable features of Hellberg's approach are (i) the treatment of both morpho-

tactic restrictions and morphophonological alternations by way of subroutines that may be used for capturing partial intraparadigmatic similarities, and (ii) the centrality of the paradigm (rather than alternation rules). There are no strict general analogues of the kinds of morphophonological alternation rules that are postulated in IP oriented models. Furthermore no abstract morphophoneme symbols or the like are used.

The present writer has designed a morphological analyzer for Finnish along similar lines, but with a more pronounced emphasis on bringing out the partial paradigmatic similarities between classes of endings (Karlsson 1985). One central idea of this paradigmatic model is that the process of morphological analysis is guided by a lexicon of concrete phoneme-level stems derived by pattern rules from an ordinary lexicon with basic forms. The pattern rules deriving the stem lexicon have as their main function to properly discriminate between productive and unproductive inflectional patterns. Thus, the integration of a new word into the lexicon is viewed as assigning to it a pattern of inflectional stems together with their appropriate morphotactic information. Once these stems have been determined by the patterns and entered into the stem lexicon, only the latter is used in analyzing forms. No rewriting rules or even automata are used in the analysis process. The pattern rules deriving the stem lexicon are of the general form <lexical phonological shape> ---> <phonological stem(s)>. All morphotactic and morphophonological generalizations are embodied in paradigms and subparadigms that flexibly may refer to one another.

6. Literature (selected)

- Y. Alam 1985 · O. Blåberg 1985 · L. Borin 1985 · B. Brodda/F. Karlsson 1980 · S. Hellberg 1972 · S. Hellberg 1978 · W. Hoepfner 1980 · W. Hoepfner 1982a · R. Kaplan/M. Kay 1981 · F. Karlsson 1985 ed. · L. Karttunen/R. Root/H. Uszkoreit 1981 · L. Karttunen/K. Wittenburg 1983 · M. Kay 1973 · M. Kay 1977 · M. Kay 1983b · M. Kay/G. Martins 1970 · R. Khan 1983 · K. Koskenniemi 1983a · K. Koskenniemi 1983b · K. Koskenniemi 1985a · K. Koskenniemi 1985b · J. Lindstedt 1985 · A. Östling 1985 · D. Reimann/B. Rüdiger 1982 · B. Rüdiger 1975 · K. Sparek-Jones 1983 · T. Winograd 1983.

Fred Karlsson, Helsinki (Finland)

22. Computational Testing of Linguistic Models in Syntax and Semantics

1. Introduction
 - 1.1. Motivations
 - 1.2. Linguistic Background
 - 1.3. Programming Languages and Computers
 - 1.4. Outline
2. Yngve's Random Generation Program
3. Transformational Grammar Programs
 - 3.1. First Systems
 - 3.2. Friedman's Transformational Model
 - 3.3. Interactive Transformational Models
4. Functional Generative Description Models
5. Programs for Montague Grammar
 - 5.1. Montague Grammar and Computational Testing
 - 5.2. Janssen's Program
 - 5.3. System of Friedman, Moran, and Warren
 - 5.4. Indurkha's Program
6. Programs for Unification Grammars
7. Conclusions
8. Literature (selected)

1. Introduction

1.1. Motivations

Computational testing of a linguistic model serves a dual purpose: it provides direct information about the model itself, and it assists in developing grammars in accord with the theory. There are some definite demands on a linguistic theory if a computational model is to be created. A theory which is generally vague cannot be modeled as a program unless an exact specification can be completed. Thus, if the initial linguistic theory is insufficiently explicit, the author of the computer model must make assumptions in order to provide an explicit form. This explicitness must be present not only in the form of the grammar, but also in the algorithms provided by the theory to analyze or to generate sentences. A linguistic theory for which a computer model exists must then be an explicit theory, although additional formalization provided in the computer modeling may fail to accord with the original intentions of the linguist.

To be of interest for computational testing, the theory must be one in which grammars can be written. A consequence of computer modeling has been to reveal the extent of the difficulties involved in setting down an explicit grammar according to the theory. An obvious but important point is that the model must contain mechanisms so that the details

of rules and the interactions of rules can be expressed.

Given that the theory can be modeled, the computational model then can serve a second function, that of grammar tester. Grammar writing is difficult because of the large number of details to be considered at each step. Not only must each individual rule perform correctly in isolation, but the rules must work together as a grammar. Grammar writing is much more difficult than rule writing. The intricate interrelations of the individual rules of a grammar make grammar writing a complex and error-prone process, much like computer programming. Unexpected consequences can arise from the interactions of rules which perform as intended in isolation. This is particularly true for models that treat both syntax and semantics, because there is an additional set of interactions to be considered.

After a grammar has been shown to work, in the sense that each rule works and the rules interact correctly, the most interesting question can be examined. Does the grammar correctly account for the facts of the language? Computer generated data can be valuable evidence toward a conclusion.

This article surveys work done between 1960 and 1986 on computer programs designed to enable their users to experiment with linguistic models in syntax and semantics. The programs were mainly developed because of the sense that modern generative grammars, regardless of the particular version of generative grammar, are sufficiently complex that a computer program offers ways of observing facts about the grammar that are otherwise unavailable. By writing grammars in the form provided by the theory, and examining the sentences they accept or generate, new insights are offered to the linguist.

In some cases, the programs have been claimed to validate a theory, that is, to show that the model is indeed adequate to the aspects of a language it attempts to cover. In other cases, the result is not so much that the theory is substantiated, but that the program provides a means to develop and gradually improve grammars within the theoretical framework, displaying at the same time the strengths and weaknesses of the theory. Com-