WIS-CS-180-73
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

Received: April 30, 1973

AUTOMATIC INFERENCE OF SEMANTIC DEEP STRUCTURE RULES IN GENERATIVE SEMANTIC GRAMMARS

Sheldon Klein

Technical Report #180

May 1973

SUMMARY

AUTOMATIC INFERENCE OF SEMANTIC DEEP STRUCTURE RULES IN GENERATIVE SEMANTIC GRAMMARS

Sheldon Klein

Computer Sciences Department
1210 W. Dayton St.
University of Wisconsin
Madison, Wisconsin 53706
U.S.A.

Techniques and methodology for the automatic inference of semantic deep structure rules in generative semantic grammars. The key conceptual devices include a representation of semantic deep structure in the notation of a 4-dimensional network with properties of at least the 2nd-order predicate calculus, and also in the notation of a compiler-driven behavioral simulation language that describes and modifies the linguistic and extra-linguistic conceptual universe of speakers. The system is able to make grammatical-semantic inferences within the frameworks of all current generative semantic linguistic models, including the case grammar of Fillmore, the presuppositional model of Lakoff, and the 1972 semantic theory of Katz.

AUTOMATIC INFERENCE OF SEMANTIC DEEP STRUCTURE RULES IN GENERATIVE SEMANTIC GRAMMARS

Sheldon Klein

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53706
U. S. A.

Introduction

This paper represents something unique in linguistics and computational linguistics in that it introduces a global linguistic theoretical model of such logical complexity that it can only exist as a computer model. This model is capable of assimilating and representing the formulations of all current linguistic generative theories, including those of generative semantics. It is capable of incorporating rules of complex discourse and socio-linguistic usage, and above all, it includes a model and methodology for inference of all the rules and data in the system.

The primary purpose of this paper is to explicate the inference of the semantic deep structure rules in the model. Of necessity, the topic must include an explication of the total model and system. However, only an abbreviated sketch sufficient for comprehension of the primary topic is presented in this paper.

Historical Background

The first successful programs for learning context-free phrase structure grammars and transformational grammars were developed by Klein and co-workers, and are described in Klein,1967, Klein et al,1967,1968, and Klein & Kuppin, 1970. The phrase structure learning program was demonstrated live, to all volunteers

* Research sponsored by National Science Foundation Grant No. GS-2595, and the Wisconsin Alumni Research Foundation.

at the December 1967 Meeting of the Linguistic Society of America, in Chicago, via a teletype linkage to a B5500 computer in Madison, Wisconsin. The programs were also demonstrated via a similar teletype link in 1968: at Carnegie-Mellon University, UCLA and USC; and in 1969 at the University of California, Berkeley.

Other researchers developed inference programs for context-free phrase structure grammars somewhat later. These include Feldman et al, 1969, Horning, 1969, Crespi-Reghizzi, 1970 and Wharton, 1973 (the list is not exhaustive).

Work involving automated semantic networks includes the early dependency approximations of Klein, 1965a,b, Klein et al, 1966; Quillian, 1966, Schank, 1969, 1972, Mel'chuk, 1970, 1972, Simmons, 1970, 1972, and Klein et al, 1971, Klein, 1972 (the list is not exhaustive). Morphological analytic work includes Klein & Dennison 1971.

Work involving variants of the 1st order predicate calculus as part of the semantic base component in natural language systems includes McCawley, 1968, Bach & Harms, 1968, Lakoff, 1969, Green & Raphael, 1969, Coles, 1969, and Petöfi, 1973 (the list is not exhaustive).

Work involving natural language compiling either into semantic representations, inference languages or simulation languages includes Kellogg, 1968, Klein et al, 1971, Klein, 1972, Heidorn, 1972, Simmons (in preparation), as well as Green & Raphael ibid and Coles, ibid. (Again the list is not exhaustive.)

Components of the System

All the components of the system except the inference mechanisms for generative semantic grammars are programmed in FORTRAN V and partially operational on a UNIVAC 1108 computer, Klein et al, 1971, Klein, 1972, Klein et al, 1973.

The key conceptual devices include a representation of the semantic deep structure in the notation of a 4-dimensional network with properties of at least the 2nd order predicate calculus, and also in the notation of a compiler driven behavioral simulation language that describes and modifies the linguistic and extra-linguistic conceptual universe of speakers. The ultimate power of the system arises from the use of constructs in the semantic deep structure network to generate new rules in the behavioral simulation language, which then control new conceptualizations and behavior of the speaker. The methodology draws upon meta-compiler theory. These features enable the system to assimilate and represent all current generative semantic linguistic theoretical models, including the case grammar of Fillmore, the presuppositional model of Lakoff, and the semantic theory of Katz, 1972.

Semantic Network

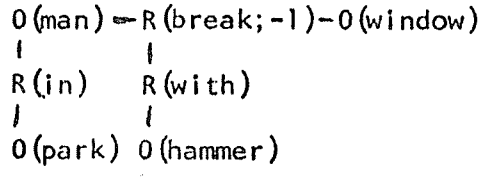
The semantic network consists of objects and relations linking those objects. The object nodes and relations have no names in themselves, only numbers. But they are linked to lexical expression lists that contain lexical variants as well as other expression forms. In examples of semantic network representations of deep structures bracketed lexical items selected from the associated lexical lists are provided with the objects and relations for convenience in reading.

As an example consider the discourse:

"The man in the park broke the window with a hammer."

"John knows that."

The deep structure network representation might resemble:



(where the -1 represents a time early than present)

But the actual representation of the semantic deep structure is more subtle and has properties not obvious in this example illustration. The network is actually composed of semantic triples. A semantic triple can consist or any sequence of 2 or 3 objects and relations. Every object in the system has a unique number or address. Every triple in the system also has a unique number and is also associated with its time of creation. The network is actually stored in the form of a hash table, wherein the actual semantic network is implied and computable rather than overtly listed. The time of creation of each triple makes the application of tense transformations easy: the simulation system maintains a clock representing 'now'. Accordingly the relative time sequence among deep structure triples is readily computable, and serves as data for generation of surface structure expression of tense, etc. The actual representation of this sentence is closer to:

1. O(man)- R(break,-time) - O(window)

R(break,-time)- R(with)- O(hammer)

2. O(man) -R(in) -O(park)

where the second triple in 1.is not actually listed separately; multi-place predicates are indexable through the primary triple.

It is worth repeating that the objects and relations are actually numbered locations with links to other objects and relations. They contain no associated content expression form other than what appears on their lexical expression lists that are also linked to them. However, a lexical expression list may contain other data than just pointers to lexical stems in a dictionary. These items include semantic triples that are not in the network (for expression of idomatic type structures) and pointers to triples that are in the network.

The objects and relations in these triples have their own links to their own lexical expression lists. The lexical expression list of an object or a relation may contain pointers to triples in the network that include triples of which it is a member.

Consider now the second sentence of the sample discourse:

"John knows that"

encoded in the semantic network as,

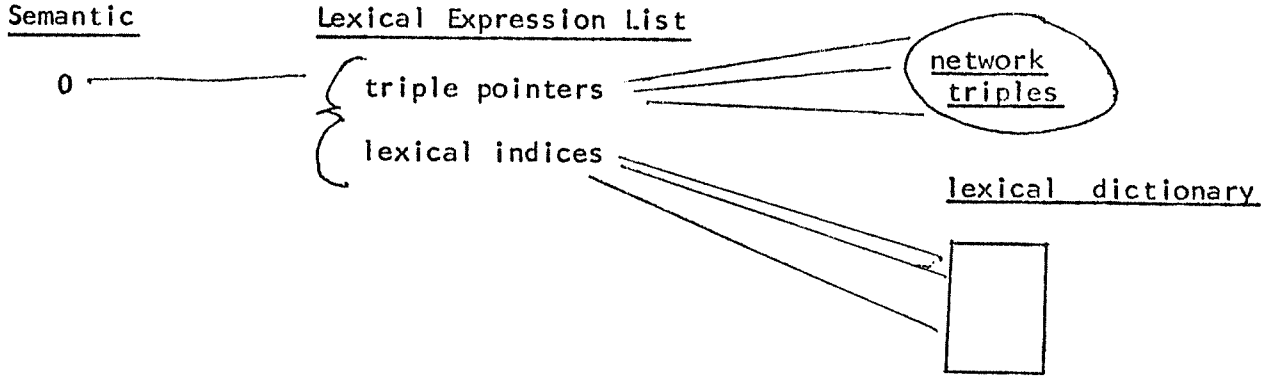
3. O(John)- R(know)- O(that)

The O(that) is a complex predicate object. Its lexical expression list contains pointers to semantic triples 1 and 2. The representation could be self-referential; if the lexical expression list of O(that) contained a pointer to triple 3, the network would represent a message approximating:

"John knows that he knows that the man in the park broke the window with a hammer."

This feature helps to give the system the logical power of the 2nd order predicate calculus (at least). Complex logical predications are represented with such predicate nodes linked by logical connective relations. Thus the statement , if A then B, where A and B are complex bodies of semantic discourse representing large portions of the semantic network, is represented simply as, O(A)- R(implication)- O(B), where O(A) and O(B) each point to lists of semantic triples that may also be of the same time--predications linking predicate objects that have pointers to triples on their lists. (Always these lists may contain self-referential pointers--serving to justify the claim that the system has the power of at least the 2nd order predicate calculus.) (Other logical devices involving classes of objects and quantifiers are associated with the simulation language manipulates and modfies the semantic network.)

A final schematic of the relevant data structures:



Generative Rules: surface structure // semantic network

The phrase structure rules in the system are part of more complex rules that compile the semantic deep structure network from surface structure--and which also serve the function of generating surface structure from the network. The general form of such a rule is:

phrase structure rule // canonical form of semantic triple

where the phrase structure rules are of the usual sort, where linked mappings between nodes in the right half of the phrase structure rules and elements in the network specification are indicated. Strictly speaking the network specification need not be limited just to a semantic triple, as will be seen in the section on inference of rules. Some examples of rules:

S→ NP VP // 0 - R

VP→ V NP // R - 0

NPP→ adj NPP // 0 -R(attribute)- 0

Note that items may occur on either side of the // marks that are not linked to items on the opposite side.

Full comprehension of these rules can best be obtained through an example of generation of surface structure from deep structure. Generalized mechanisms

for context sensitive rules and transformations are part of the model. But they are of a type more basic and primitive than in most exisiting linguistic generative models. They can represent more complex types of transformations when properly combined.

A Generation Example

Assume a grammar containing the following surface//semantic rules:

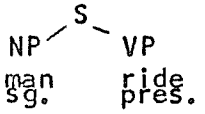
1. S→ NP VP // 0 - R
2. NP→ NP PP // 0 - R
3. NP→ Det NPP // 0
4. NPP→ adj NPP // 0 - R - 0
5. NPP→ terminal
6. VP→ VPP PP // R - R
7. VPP→ V NP // R - 0
8. VPP→ terminal
9. V→ terminal
10. PP→ prep NP // R - 0
11. prep→ terminal

Assume that the semantic deep structure triple set to be used in the generation is:

0(man)- R(ride) -0(bicycle)
R(ride)- R(in) - 0(park)
0(man)- R(is)- 0(tall)

The overlap of various objects and relations in more than one triple is known to the generator by various link markings. The time associated with each triple is also part of the data. A starting symbol S is selected. A prior selective mechanism has placed the triple representing the main predication of the sentence at the top of the triple list. The generative component inspects all S rules whose right hand network description is of the same canonical form as that of the first semantic triple. Here the condition is not satisfied by the only S rule, 1. The triple is then broken into two overlapping parts, 0(man) - R(ride) and R(ride) - 0(bicycle). The S rules are then inspected for matches with the fractioned canonical forms. The first matches rule 1.

At this point lexical stems are selected from the lexical expression lists associated with the objects and relations in the matched triple fraction. A selected lexical item is tenatively assigned to the node indicated by the link in the syntactic//semantic rule. Grammatical information associated with the lexical item in the dictionary indicates whether or not it can serve as the head of a construction dominated by the node under which it was selected. In this case:

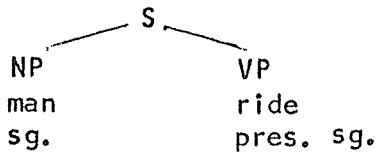


Lexical Dictionary

	NP	VP	PP	ADV
man	1	1	0	1
ride	1	1	0	0

A bit vector in the dictionary indicates the applicability of a particular node. Note that both man and ride could serve as nouns or verbs. The grammar also marks the forms when appropriate for application of low level transformations at a later stage. If man were selected as a stem to fill a slot defined by an adverb node, ADV, it would at this time be marked for later application of a transformation that would add -ly to it. If the lexical dictionary should prevent the selection of a form, an alternate from the lexical expression list is tried. If none on the list are acceptable, another surface//semantic rule is selected to express the semantic triple. Number for objects is indicated directly in the lexical expression list associated with the particular object (some objects may be inherently plural, as in the case of objects that represent classes). As soon as the lexical items are selected and accepted (the stage in the preceeding diagram), a test for applicability of a high level transformation is made. This transformation uses as its index information that never becomes more complex than the subtree indicated

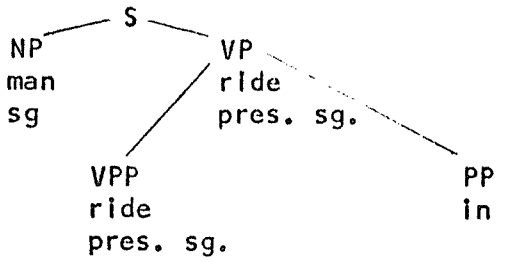
in the above diagram--"a nuclear family tree"- a parent node and its immediate descendents. Often, as in this case, the lexical items are not relevant to the transformation, that here marks the VP with the same number as the NP.



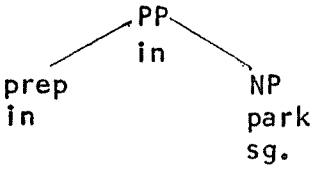
Low level transformations that operate only on terminals and their immediate parent nodes will actually convert the stems to the appropriate words at the end of the generation process. The transformation markings supplied by the high level transformations are carried with the lexical items and may serve as part of the data for defining the applicability of other high level transformations. This breaking up of the transformational component into two types of limited environment primitive operations permits extremely rapd transformational generation and parsing algorithms. The complex labor of searching for applicable environments common to most other automated transformational systems is avoided.

Tense information is obtained from the time marking of the triple. The simulation system maintains a clock, and the relative time order of the triples in the deep structure generation list can be computed, so that the proper items may be marked for application of transformations handling tense.

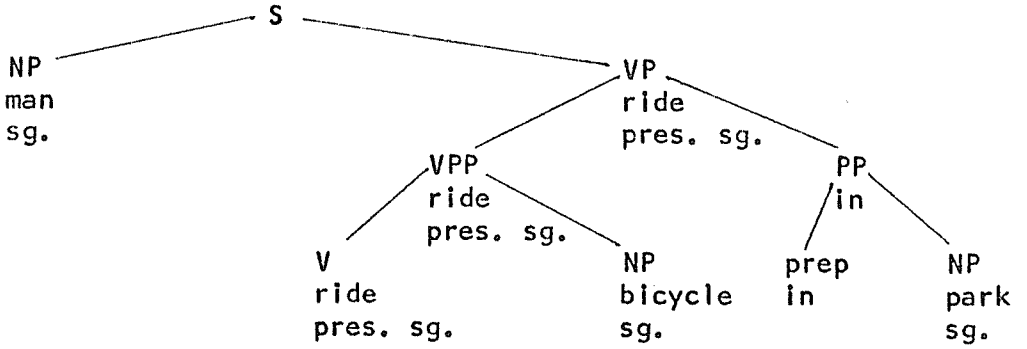
Continuing the generation process, the system saves the remainder of the first triple and skips to the second because of a special link between their relations indicating simultaneity. No VP rule matches the second triple, and it is split into the fractions R(ride) - R(in) and R(in) - R(park). The first fraction matches rule 6. After lexical item in is selected, the tree appears as:



The second triple fraction matches rule 10, yielding after lexcal selection:

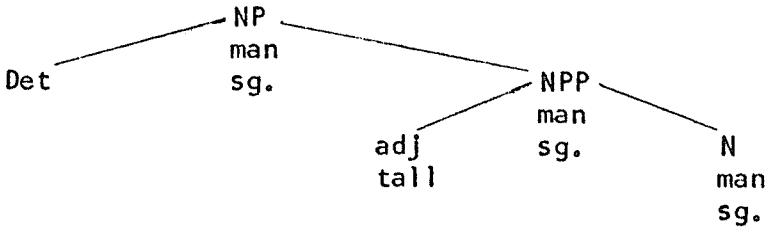


At this point, the second fraction of the first triple is matched against rule 7, and, after lexical selection, the entire tree appears as:



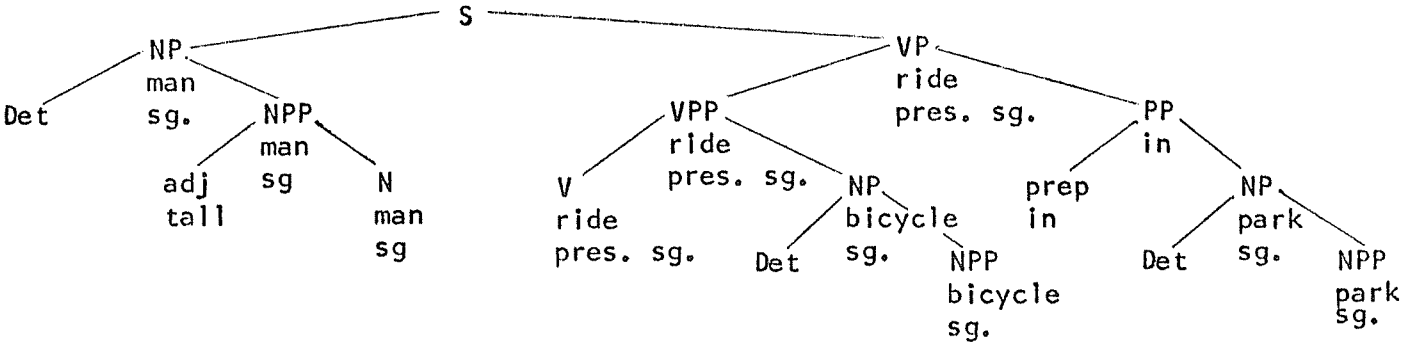
No rule matches the remaining triple O(man) - R(is) - O(tall). Rule 2 matches the first fraction, but the lexical list for the relation R(is) contains no item acceptable as a PP node descendant. Accordingly, rule 3 is selected. At this point a high level transformation marks the Det for conversion to an appropriate form at the final stage. (If the lexical item had been a proper noun, the Det node would have been marked for deletion.)

At this point rule 4 applies to the entire, unfractionalized, remaining triple, yielding the subtree:



At this point rule 3 is applied to the NP nodes dominating bicycle and park.

The resultant tree is:



The final, low level transformations are applied, yielding the sentence:

"The tall man rides the bicycle in the park"

Note that the semantic triple set might have generated more than one sentence to express the content--either by deliberate stylistic design, or because the rules might not have permitted a grammatically correct construction incorporating the entire semantic structure.

Semantic Parsing

The recognition process is the inverse of the generation process. Exactly the same rules are used, but they are applied in the reverse direction. The surface structure string is reduced to lexical stem variants. Each lexical entry in the dictionary also contains a back pointer to each semantic object or relation that might contain it on its lexical expression list. In general, the mappings are complex:

Semantic objects and relations: x y z

Lexical items a b c d e f

Ambiguities arise as a matter of course in the parsing process. But the surface structure// semantic network compiler rules are embedded in a larger system that includes a semantic network description of a universe of discourse, and potentially, a universe representing the semantic world of a speaker. The disambiguation process may use this information as well as the rules of socio-cultural behavior encoded in the simulation language. The amount and kind of data available for ambiguity resolution is greater than that in any current automated or theoretical linguistic model, for it can even include information about physical environment and universe of the speaker.

The Behavioral Simulation Language

The system contains another surface structure semantic network grammar. But this grammar is for the generation of sentences in the behavioral simulation language--sentences that are rules governing the modification of the semantic network itself--written in a language that is compiled into executable program code by yet a third grammar-- that of the simulation language compiler. The system is described in Klein et al, 1971, Klein, 1972 and Klein et al, 1973.

The same semantic deep structure is used to generate in natural languages and in the simulation language. Thus it is possible to represent the rules of socio-cultural behavior descriptively in the semantic network, and to generate discourse about them in a natural language. At the same time, the rules of the simulation can be used to create new semantic network structures that can then be used to generate brand new simulation rules, etc. . This feature provides a natural language meta-compiler capability and completes the claim that the system has the logical power of at least the 2nd order predicate calculus.

Rules in the simulation language enter and delete semantic triples from the network. Deleted triples remain, but are associated with a time of termination as well as a time of creation. The rules may enter semantic triples subject to complex logical conditions pertaining to the existence or non-existence of other semantic triples already in the network. The conditions may be probabilistic or deterministic. The rules may refer to classes of objects, including objects that are themselves complex predicate variables. They are capable of encoding contary-to-fact,belief information without confusion with the accepted reality of the network. (e.g. "John thinks that A" where A is contrary to fact). Negation is handled in various ways--either as just indicated, or computed from the existence of a termination time for a triple.

These rules of behavior may be learned through verbal description from an informant. Information provided by an informant that involves objects and relations not in the network can stimulate a process that enters new objects and relations into the data base. (Facilitating the modelling of presuppositional grammatical theories)

INFERENCE OF GENERATIVE SEMANTIC RULES

Everything in the preceeding may be treated as introduction to the following exposition. The germ of the inference techniques have been described in Klein, 1967, Klein et al, 1968, Klein and Kuppin, 1970.

At this point a comment on the use of a Universal Semantic Component is nessary. For inference purposes it is easier, but not necesseary, to assume such a component. Techniques can be described for inferring a non-universal, arbitrary semantic component. However, in this exposition, I shall assume a universal semantic component that is assumed to be common to the informant and the grammatical inference device. This is equivalent to the assumption that

there exists a program for converting English translations of productions in the language under analysis into their deep structure representation. It is not important for the exposition of the learning mechanisms whether or not a good or correct universal semantic component is used--only that it be common to both the informant and the inference device. But even this assumption is not entirely necessary; even with the initially assumed common component it is possible to introduce the learning of additional deep structure that is not common to both informant and the inference device (an indication that the assumed universal component was faulty).

An Analytic Example (hand simulated solution)

The following example uses an amended version of a problem in Koutsoudas, 1966.

Problem 59: Japanese

1. nara wa hanasi o kiita
   Nara heard the story.

1a. (added) jon wa hanasi o kiita
   John heard the story.

1b. (added) nara wa jon o kiita
   Nara heard John.

2. anata wa nara o kiita
   You heard Nara.

3. jon wa hanasi o sinzita
   John believed the story.

4. nara wa jon o sinzita
   Nara believed John.

5. anata wa nara ga hanasi o kiita to sinzita
   You believed that Nara heard the story.

6. nara wa jon ga hanasi o sinzita to kiita
   Nara heard that John believed the story.

7. jon wa siawase da
   John was happy.

7a. (added) daigakusei wa siawase da
   The student was happy.

8. anata wa jon ga siawase da to kiita
   You heard that John was happy.

9. nara wa daigakusei da
Nara was a student.

10. jon wa nara ga daigakusei da to sinzita
John believed that Nara was a student.

11. nara wa anata ga daigakusei da to kiita
Nara heard that you were a student.

12. daigakusei wa nara ga siawase da to sinzita
The student believed that Nara was happy.

13. jon wa anata ga siawase da to kiita
John heard that you were happy.

But not:

*hanasi wa nara o kiita
*hanasi wa siawase da
*jon wa hanasi da
*anata wa jon ga kiita to siawase da
*jon wa nara ga kiita to daigakusei da

This hand simulation of an analysis will avoid some powerful devices available to the system. Remember that the semantic deep structure of the English glosses is assumed to be available. These could be used to generate queries in English far simpler forms than the complex constructions offered in the example. Except for the added inputs such aids are not used.

The notation for the rules coined deserves comment. Each rule begins with the letter S followed by a number. The name of a rule is the complete alpha-numeric descriptor. This convention is used because it conforms somewhat to a similar convention used in earlier writings on grammatical inference (Klein et al, ibid), and facilitates comparisons with earlier work.

After input sentence 1, the following partial rule is formulated:

1. S1→ nara wa hanasi o kiita // O(Nara) - R(hear) - O(story)

(tense is ignored as their is no contrast in the problem)

Input sentence 1a at first yields the rule:

2. S2→ jon wa hanasi o kiita // O(John) - R(hear) - O(story)

But a double mismatch

in otherwise identical rule structures permits the following grammar revision:

1. S1→ S2 wa hanasi o kiita // 0 - R(hear) - 0(story)

2. S2→ terminal

and the addition to the lexical dictionary :

	S2
nara	1
jon	1

On the basis of this embryonic grammar, input 1b parses to:

S2 wa S2 o kiita // 0 - R(hear) - 0

The information permits matching with rule 1 and the identification of 'hanasi' asa lexical expressant of 0(story)

The tenative grammar:

1. S1→ S2 wa S2 o kiita // 0 - R(hear) - 0

2. S2→ terminal

Lexical Dictionary

	S2
nara	1
jon	1
hanasi	1

is posited and then rejected after it is used to produce the test sentence: "hanasi wa jon o kiita" which is rejected by the informant.

Accordingly, the equivalent of a sub-category is created,yielding the grammar:

1. S1→ S2 wa S3 o kiita // 0 - R(hear) - 0

2. S2→ terminal

3. S3→ terminal

Lexical Dictionary

	S2	S3
nara	1	1
jon	1	1
hanasi	0	1

Ignore input sentence 2 for the moment and accept instead input 3, which parses to:

S2 wa S3 o sinzita // 0 - R(believe) - 0

When compared with rule 1, the mismatch kiita/sinzita is obtained, permitting their identification as lexical expressents of 0(hear) and R(believe). A class S4 is created, and the forms are added to the lexical dictionary yielding:

1. S1→ S2 wa S3 o S4 // 0 - R - 0
2. S2→ terminal
3. S3→ terminal
4. S4→ terminal

Lexical Dictionary

	S2	S3	S4
nara	1	1	0
jon	1	1	0
hanasi	0	1	0
kiita	0	0	1
sinzita	0	0	1

At this point a comment on pronouns is warranted. A pronoun may be on the lexical expression list of any number of objects. Its precise form is subject to modification by transformation. The form is entered in the lexical dictionary in the inference process provisionally in a full form. If variants are encountered, the lexical dictionary may list only one or a few, with transformations handling other variants. Thus the only impact on the grammar caused by input sentence 2, is the addition of the lexical dictionary entry:

	S2	S3	S4
anata	1	1	0

Input sentence 4 is completely parsed as far as the current grammar is concerned--reducing to: S2 wa S3 o S4 // 0 - R - 0 which is identical with rule 1.

More interesting things happen when input sentence5 is added. At first it yields the partial parse:

S2 wa S3 ga S3 o S4 to S4 // 0(you) - R(believe) - 0(predicate object)
0(Nara) - R(heard) - 0(story)
(or more precisely, 0 - R - 0
0 - R - 0

This is added as rule S5 , but a rejected test production would force the first S3 to be changed to S2, yielding:

5. S5→ S2 wa S2 ga S3 o S4 to S4 // 0 - R - 0
0 - R - 0

At this point a new type of semantically based heuristic is used. The mappings from the syntactic side of S5 to the two linked semantic triples permit a powerful syntactic//semantic decomposition and matching. On the basis of the condition that the syntactic string, S2 ga S3 o S4, contains mappings only to a single semantic triple, that string and the relevant triple are extracted to form the rule S6, yielding the changes:

5. S5→ S2 wa S6 to S4 // 0 - R - 0

6. S6→ S2 ga S3 o S4 // 0 - R - 0

At this stage rule 6 is matched against rule 1, yielding the discrepancy, wa/ga . It is now possible for the system to make a chain of complex inferences, which if valid, permit a global solution to the problem, but which if invalid permit the analysis to continue at a less general stage. (The system always maintains its data in such a way that tenative complex inferences may be undone, and earlier states of the grammar restored.)

An attempt is made to combine rule 6 and rule 1. A commitment to coin a transformation accounting for the wa/ga discrepency is noted. Rule 6 is deleted, but now the occurence of the S6 node in rule 5 must be replaced by S1:

1. S1→ S2 wa S3 o S4 // 0 - R - 0

.

.

5. S5→ S2 wa S1 to S4 // 0 - R - 0

Two high level transformations are posited to account for the wa/ga variation:

T1. S5 (S2 wa S1 to S4) → S5 (S2 wa S1^{T2} to S4)

T2. S1^{T2} (S2 wa S3 o S4) → S1 (S2 ga S3 o S4)

Note that if there are other environments that also yield a wa/ga variation, the process might be divided with the low level transformations. In that case T2 would merely flag 'wa' for later application of a low level transformation to convert it to 'ga.'

The remaining inputs have a major effect on the grammar. Skipping input sentence 6, and entering instead, input sentence 7, yields the rule:

6. S6→ S2 wa siawase da // 0 - R(is) - 0(happy)

Input sentence 7a has only the effect of adding 'daigakusei' to the lexical dictionary (after verifying test productions):

	S2	S3
daigakusei	1	1

The addition of input sentence 9 (skipping 8) permits the revision of rule 6:

6. S6→ S2 wa S3 da // 0 - R - 0

and the addition of lexical entry:

	S2	S3
siawase	0	1

(only after appropriate testing with implied test sentences)

An attempt is now made to combine rule 6 and rule 1 on the grounds that the rules are identical except for 'o', and that this 'o' is without any link to the semantic portion of the rule. (The absence of 'o' would imply the addition of 'da' to S4.) The resultant changes affect rules 1 and T2, replace rule 6, add rule 7, and also enter 'da' in the lexicon, yielding the final grammar:

1. S1→ S2 wa S3 S6 // 0 - R - 0
2. S2→ terminal
3. S3→ terminal
4. S4→ terminal
5. S5→ S2 wa S1 to S4 // 0 - R - 0
6. S6→ o S4 // R
7. S6→ terminal

Lexical Dictionary

	S2	S3	S4	S6
nara	1	1	0	0
jon	1	1	0	0
hanasi	0	1	0	0
kiita	0	0	1	0
sinzita	0	0	1	0
daigakusei	1	1	0	0
siawase	0	1	0	0
da	0	0	0	1

High Level Transformations

T1. S5 (S2 wa S1 to S4) → S5 (S2 wa S1 ^{T2} to S4)

T2. S1 ^{T2} (S2 wa S3 S6) → S1 (S2 ga S3 S6)

This grammar is now sufficient to account for all the remaining input sentences. Consider input 8. By phrase structure rules this parses to:

S2 S2 S3 S6 S4
anata wa jon ga siawase da to kiita

By the inverse of T1 and T2, this reduces to:

S2 wa S2 wa S3 S6 to S4

then by phrase structure rule 1 to:

S2 wa S1 to S4

which reduces to S5 (by rule 5), completing the parse.

All the remaining inputs have identical parses.

References

Bach, E. & Harms, R. T. 1968. Nouns and noun phrases. In Universals in Linguistic Theory. Chicago: Holt, Rinehart and Winston, Inc.

Coles, S. L. 1968. An on-line question-answering system with natural language and pictorial input. In Proc. ACM 23rd Nat. Conf. Princeton: Brandon Systems Press.

Crespi-Reghizzi, S. 1970. The mechanical acquisition of precedence grammars. Report UCLA-ENG-7054, School of Engineering & Applied Sciences, UCLA.

Feldman, J. A., Gips, J., Horning, J. J., & Reder, S. 1969. Grammatical Complexity and Inference. Tech Report No. CS 125, Computer Science Department, Stanford University.

Green, C. C. & Raphael, B. 1968. Research on intelligen question-answering systems. In Proc. ACM 23rd Nat. Conf., Princeton: Brandon Systems Press.

Heidorn, G. E., 1972. Natural language inputs to a simulation programming system. Report NPS-55HD72101A, Naval Postgraduate School, Monterey California

Horning, J. J. 1969. A study of grammatical inference. Tech Report No. CS 139, Computer Science Department, Stanford University.

Katz, J. 1972. Semantic Theory. New York: Harper & Row.

Kellogg, C. H. 1968. A natural language compiler for on-line data management. In Proc. AFIPS 1968 FJCC, Vol. 33, Montvale: AFIPS Press.

Klein, S. 1965a. Automatic paraphrasing in essay format. Mechanical Translation, Vol. 8, Nos. 2 & 3 combined.

Klein, S. 1965b. Control of style with a generative grammar. Language, Vol. 41, No. 4.

Klein, S. 1967. Current research in the computer simulation of historical change in language. Actes du X^{e} Congrès International des Linguistes, Bucharest 1967. Vol. IV. (actually published 1970). Academy of the Socialist Republic of Roumania.

Klein, S. 1972. Computer simulation of language contact models. UWCS Tech Report 167. Also in press, Proceedings SECOL 8 Conference, Oct. 26-29, 1972, Georgetown Univ.

Klein, S., Aeschlimann, J. F., Balsiger, D. F., Converse, S. L., Court, C., Foster, M., Lao, R. Oakley, J. D., & Smith, J. 1973. Automatic novel writing: a status report. In press, Proceedings of the International Conf. on Computers in the Humanities, July, 1973.

Klein, S., Davis, B, Fabens, W., Herriot, R., Katke, W., Kuppin, M. A., & Towster, A. 1967a AUTOLING: an automated linguistic fieldworker. Second International Conference on Computational Linguistics, August 1967, Grenoble.

Klein, S., Fabens, W., Herriot, R., Katke, W., Kuppin, M. A., & Towster, A. 1968. The AUTOLING system. UWCS Tech Report 43, Univ. of Wisconsin Computer Sci. Dept.

Klein, S. & Dennison, T. A. 1971. An interactive program for learning the morphology of natural languages. UWCS Tech Report 144. Also in press, Proceedings of the 1971 International Conf. on Computational Linguistics, Debrecen, Hungary, September 1971. , Mouton & Hungarian Academy of Sciences.

Klein, S., Lieman, S. L. & Lindstrom, G. D. 1966. DISEMINER: a distributional-semantics inference maker. Tech Report of Carnegie Inst. of Tech., and, Computer Studies in the Humanities & Verbal Behavior, Vol. 1, No. 1, Jan. 1968.

Klein, S., Oakley, J. D., Suurballe, D. A., & Ziesemer, R. A. 1971. A program for generating reports on the status & history of stochastically modifiable semantic models of arbitrary universes. UWCS Tech Report 142. Also in Statistical Methods in Linguistics 8, 1972. Also in press, Proc. of 1971 Int. Conf. on Computational Linguistics, Debrecen, Hungary, September 1971, Mouton & Hungarian Academy of Sciences.

Klein, S. & Kuppin, M. A. 1970. An interactive, heuristic program for learning transformational grammars. UWCS Tech Report 97. Also in Computer Studies in the Humanities & Verbal Behavior, Vol. 3, No. 3, October 1970.

Koutsoudas, A. 1966. Writing Transformational Grammars. New York: McGraw-Hill.

Lakoff, G. 1969. Generative semantics. In Semantics--An Interdisciplinary Reader in Philosophy, Linguistics, Anthropology and Psychology. London: Cambridge Univ. Press.

McCawley, J. D. 1968. The role of semantics in a grammar. In Universals in Linguistic Theory, Bach & Harms, editors, Chicago: Holt, Rinehart and Winston Inc.

Mel'čuk, I. A. 1972. Уровни представления высказываний и общая структура модели «Смысл ⟷ Текст». Предварительные Публикации, Выпуск 30, Институт Русского языка АН СССР. Москва.

Mel'čuk, I. A. & Žolkovskij, A. K. 1970. Toward a functioning 'meaning-text' model of language. Linguistics 57, May 1970.

Petöfi, J. S. 1973. Towards an empirically motivated grammatical theorie of verbal texts. Bielefelder Papiere zur Linguistik und Literaturwissenschaft. Also in press, in Studies in Text Grammar, Petöfi & Rieser, editors, Dordrecht: Reidel.

Quillian, R. 1966. Semantic memory. Ph. D. Thesis, Carnegie-Mellon University, Pittsburgh. Also, Cambridge, Mass.: Bolt, Beranek & Newman, 1966.

Schank, R. C. 1969. A conceptual dependency representation for a computer-oriented semantics. AI Memo-83, Computer Science Department, Stanford University.

Schank, R. C. 1972. Conceptual dependency: a theory of natural language understanding. Cognitive Psychology, Vol. 3, No. 4.

Simmons, R. F. 1970. Some relations between predicate calculus and semantic net representations of discourse. Computer Science Dept. Preprint, U. of Texas, Austin.

Simmons, R. F. 1972. Generating English discourse from semantic networks. Communications of the ACM, Vol. 15, No. 10.

Simmons, R. F. (in preparation). Compiling semantic networks from English sentences.

Wharton, M. R. 1973. Grammatical inference and approximation. Tech Report 51, Computer Science Dept., Ph. D. Thesis, University of Toronto.