CASE STRUCTURE SYSTEMS

Bertram Bruce
Computer Science Department
Rutgers University
New Brunswick, New *Jersey,* U. S. A.

Abstract

A logic for case structure systems is presented which allows variations in the order and number of terms in atomic formulas. This logic is used to describe and characterize four existing case systems. A computer program which allows flexible case structures is then described. Applications of the program to medical record analysis and disease modeling are used to illustrate important concepts about case systems. Several open problems are also discussed.

1. Introduction

Much of the research in natural language processing has focused on the problem of storage structures, or the question, "How is the information or meaning of a sentence to be represented once a sentence has been parsed?" Some of the storage structures have been special purpose organizations such as the SAD SAM program of Lindsay 12, the BASEBALL program of Green, et al [8], *or the* STUDENT program of Bobrow[1]. Others have been semantic networks such as TLC by Quillian[15], PROTOSYNTHEX III by Simmons[2]^, MEMOD by Norman and Rumelhart[14], or MENTAL by Shapiro[19], still others have been based on dependency grammar such as Schank's conceptual parser-*[7] or on predicate calculus, such as QA3 by Green and Raphael[7] or PCDB by Sandewall[16],

A common feature of several of these natural language processors is a recognition of deep case relations as discussed by Fillmore[6]. Some systems which have not explicitly used a case structure have nevertheless used case-like mechanisms. Jn addition, case-like systems have been used in modeling in medicine and psychology, even without natural language.

It appears now that a formalization of case structure languages would be useful and feasible. On the one hand the large amount of data on case systems appears to be structured in similar ways and yet differences in terminology and approach have created serious communication problems. On the other hand, even if we admit that the usual first order logics somehow capture the essential nature of these systems, the simplifications imposed there for elegance and theoretical usefulness make them less useful as a means for examining and comparing case *structure systems.*

This paper presents (in section 2) a language for case structure systems which is built on first order logic and yet allows such things as omitted and rearranged terms in atomic formulas, prepositions, and the checking of case restrictions in the determination of well-formedness. The case logic should make it significantly easier to formulate, analyze or compare case structure systems.

this research, was supported by the National Institutes of Health. Resource on Computers in Biomedicine at Rutgers University (Grant RR 643)

In section 3 a few representative systems with cases are expressed in the formalisms of case logic. Section *4* contains a description of the CHRONOS 11 program[3]. This program is unusual in having a flexible case structure. For example it has been used with one case structure for the analysis of medical records and with another as a model for glaucoma.

2. Case Logic

This section describes the essential characteristics of a case structure system or case logic (CL). It appears that an adequate formalization can be made within a first order framework. The differences between this and a pure predicate *calculus* involve the introduction of some fixed, or constant predicates, some additional axioms, and some syntactical changes to reflect the fact that certain languages of interest allow variations in the order and even omissions of some terms.

The axiom system for a case logic depends heavily on the specific cases which are used and, of course, on the interpretations we wish to allow. Because no axioms are given here it is perhaps more appropriate to call *CL* a "notational system" rather than a "logic". However, it should be remembered that in any particular CL *one must* include an axiom system as well as naming the various syntactic and semantic objects in the system.

We have all of the usual symbols found in first order logic:

$\mathcal{P} = \{P_i^{(n_i)}\}$ is a countable set of n_i-ary predicates,

$\mathcal{F} = \{f_i^{(n_i)}\}$ is a countable set of n_i-ary functions,

$\mathcal{X} = \{x_i\}$ is a countable set of variables,

$\mathcal{L} = \{L_i^{(n_i)}\}$ is a finite, complete set of n_i-ary logical connectives,

plus the necessary punctuation, quantifier, and bracketing symbols. In addition, case logic has some new objects:

$\mathcal{M} = \{M_i^{(1)}\}$ is a countable set of constant unary predicates, called features or (semantic) markers (cf. Chomsky[5], or Katz[10])

$\mathcal{C} = \{C_i\}$ is a finite, non-empty set of cases. There is a distinguished case, C_0, which is always in $\mathcal{C}$

$\mathcal{Q} = \{q_i\}$ is a finite (possibly empty) set of prepositions.

The sets $\mathcal{M}$ and $\mathcal{C}$ are sets of symbols in the meta-language for CL. That is, they are used to determine such things as well-formedness rather than as symbols in the object language. When we consider a natural language, such as English, where the object language is its own meta-language then it is allowed, but not necessary, that $\mathcal{M}$ be a subset $\mathcal{P}$. It is important to note that $\mathcal{M}$ consists of constant predicates. The interpretation of a feature is thus fixed to a simple unary relation, or attribute. In practice we might specify a given feature by simply checking off (as in a lexicon) the terms to which it applies.

Using these primitives we make several definitions to give eventually the rules of formation for CL.

The <u>case structure</u>, $\mathcal{Y}_{P_i^{(n_i)}}$, of a predicate $P_i^{(n_i)}$, is a set of sequences of cases $(C_1, C_2, \ldots, C_k)$ such that for each sequence, $K \leq n_i$.

The set of <u>terms</u> is the smallest set $\mathcal{T}$ such that:

(1) $x_i \in \mathcal{T}$ for each $x_i \in \mathcal{X}$

(2) $f_i^{(0)} \in \mathcal{T}$ for each $f_i^{(0)} \in \mathcal{F}$

(3) if $t_1, t_2, \ldots, t_n \in \mathcal{T}$ then for each $f_i^{(n_i)} \in \mathcal{F}$

$f_i^{(n_i)} t_1 t_2 \ldots t_{n_i} \in \mathcal{T}$

The set of <u>preposition terms</u> is the smallest set $\mathcal{T}_c$ such that:

(1) $\mathcal{T} \subseteq \mathcal{T}_c$

(2) if $t_i \in \mathcal{T}$ then $q_j t_i \in \mathcal{T}_c$ for each $q_j \in \mathcal{Q}$

Thus a preposition term is either a term in the usual first order sense or a term preceded by a preposition.

A <u>case condition</u> is a relation $R_{C_i} \subseteq \mathcal{P} \times \mathcal{T}_c$.

We assume that if case C_i is not in one of the sequences of the case structure for a predicate P then R_{C_i} (P,t) is false for each preposition term t.

The relational statement R_{C_i} (P,t) may be read "the preposition term t is allowed to serve in the case C_i with respect to P". We normally require that a case condition be defined in terms of features ($\mathcal{M}$). Then this rather semantic-like property can be used to determine well-formedness without encountering difficulties such as attempting to use derivability to define well-formedness.

As an example of a case condition, suppose that "dative" is one of the cases in a case sequence in the case structure for "give". We might require that the preposition term which serves the dative position must be "animate" where "animate" is a feature. Then

R_{dative} (*give, John*) *holds*

while R_{dative} (give, ocean) does not.

The set of <u>kernel strings</u> is the smallest set $\mathcal{K}$ such that if $t_1, t_2, \ldots, t_j \in \mathcal{T}_c$, $P_i^{(n_i)} \in \mathcal{P}$, and $j \leq n_i$ then $P_i^{(n_i)}$ $t_1 t_2 \ldots t_j$; $t_1 t_2 \ldots t_m P_i^{(n_i)} t_{m+1} \ldots t_j; \ldots$; $t_1 t_2 \ldots t_j P_i^{(n_i)} \in \mathcal{K}$

In other words, a kernel string is formed by inserting an n_i-ary predicate into a string of $j \leq n_i$ preposition terms.

A <u>case signal</u> is a function $S_i: \mathcal{K} \to 2^{\mathcal{C}}$ (the power set of $\mathcal{C}$). For each kernel string K, S_i indicates a set of possible cases for the ith preposition term. The value of S_i (K) may depend on

(1) the predicate in K
(2) the arrangement of all the preposition terms in K

and (3) the prepositions, if any, in the preposition terms.

For example, using Fillmore's [10] case system, S_3 applied to the sentence "John broke the window with Mary" might give the set {agentive, instrumental} indicating that the 3rd term ("Mary") may bear the agentive relationship to "broke" (if Mary assists John) or the instrumental relationship (if Mary is thrown by John). The distinguished case (C_0) may be used as the image of k under S_i if the predicate in k has an irrelevant case structure (see end of this section).

Let A be a kernel string with predicate P and preposition terms $t_1, t_2, \ldots, t_n$. Then A is an <u>atomic formula</u> iff for each preposition term, t_j, there is a case, C_j, such that:

(1) $C_j \in S_j$(A) (C_j is signaled by S_j)

(2) R_{C_j} (P,t_j) holds (the case conditions for case C_j, predicate P, and term t_j are satisfied)

and (3) $(C_1, C_2, \ldots, C_n) \in \mathcal{Y}_P$ (the sequence of cases in A must be an element of the case structure for P)

The set of <u>well-formed formulas</u>, $\mathcal{W}$, is then built up in the usual way out of atomic formulas, quantifiers, and logical connectives.

Thus an atomic formula in CL is a string consisting of preposition terms and a predicate, subject to restrictions that (a) each term be assigned a case, (b) the case conditions (defined by features) are satisfied, and, (c) the sequence of cases is allowed by the predicate. If any of these restrictions fail the atomic formula (or simple sentence) is then ill-formed. As we have deliberately introduced some semantics into the syntax it may be that a so-called "grammatical but semantically anomalous" sentence would be unacceptable sytactically in CL. For example, the sentence "the book sold John" is well-formed only if there is a word sense for "book" which satisfies a feature such as "animate" so that the agentive case condition can hold. In fact, we consider such a sentence to be well-formed in exactly those cases where "the book" has been personified.

It is not surprising that other kinds of semantic anomaly are allowed. For instance "2 + 2 = 5" is well-formed in CL. Some semantic ambiguity can also be removed, but, again, many ambiguous sentences will have to be considered well-formed. For example, in "Mary was seen by John", "John" may serve either an agentive or a locative relationship to "see". However, in "Mary was seen by the river" we might expect a CL system to reject the agentive interpretation for "the river". Whenever ambiguity as to case does occur then there will be more than one case for one or more case terms in a formula. Thus for each "j" in the definition of well-formed formula, above, there may be more than one "C_j" satisfying the definition.

Case signals use purely syntactic information to select a set of possible cases. Case conditions also select possible cases but use more semantic information. An interesting question, posed by William Fabens, is how these functions should be made to interact. It is not clear at this point whether one should apply case signals first or case conditions, or even whether there are situations where computational differences exist.

Corresponding to the syntactic development given above we need a semantics for CL:

A semantic structure for CL has the form:

$\mathcal{A} = [A, \{R_i^{(n_i)}\}, \{O_j^{(n_j)}\}]$ where A is a set of individuals (objects), each R_i is an n_i-ary relation, on A and each O_j is an n_j-ary operation on A.

An interpretation of the terms is a mapping $\mathcal{I}: \mathcal{T} \to A$ which assigns (in the usual way) an object in A to each $t \in \mathcal{T}$. The interpretation of preposition terms is then a mapping $\mathcal{I}_c: \mathcal{T}_c \to A$ which assigns an object in A to each preposition term, qt, as follows:

$$\mathcal{I}_c(qt) = \mathcal{I}(t)$$

The denotation of a predicate $P_i^{(n_i)} \in \mathcal{P}$ for a given case sequence $\alpha \in \mathcal{D}_{P_i^{(n_i)}}$ is a mapping D which gives a relation $R_j^{(n_j)}$ such that $n_i \geq n_j$ = the length of α. We write $D(P_i^{(n_i)}, \alpha) = R_j^{(n_j)}$.

The valuation of the formulas in $\mathcal{W}$ is a mapping $V: \mathcal{W} \to 2$ (where 2 is the Boolean algebra of two elements) defined as follows:

(1) if $\phi \in \mathcal{W}$ is atomic and $t_1, \ldots, t_j$ are the preposition terms in ϕ, then $V(\phi) = 1$ iff

$[D(P_i^{(n_i)}, (C_1, C_2, \ldots, C_j))]\ (\mathcal{I}_c(t_1), \ldots, \mathcal{I}_c(t_j))$ holds where the C_k's are the cases of the preposition terms in ϕ.

(2) Otherwise V is computed in the usual way.

It is worthwhile noting that while CL has several unusual characteristics, appropriate restrictions allow the reproduction of a standard first order language within CL. For example, let $\mathcal{M}$ and Q be empty and let $\mathcal{C} = \{C_0, C_1\}$. Assign the case structure $\{(C_0, \ldots, C_0)\}$ (a sequence of n C_0's) to each n-ary predicate. For case conditions let $R_{C_0} = R_{C_1} = \mathcal{P} \times \mathcal{T}_c = \mathcal{P} \times \mathcal{T}$ (since Q is empty $\mathcal{T}_c = \mathcal{T}$). For any kernel string K and any i, the case signal S_i is defined as follows:

$$S_i(k) = \begin{cases} C_0, & \text{if } t_i \text{ is the ith term to the right of P} \\ C_1, & \text{otherwise} \end{cases}$$

Then the set of atomic formulas is simply the set of those kernel strings which have n terms for an n-ary predicate, all in the usual order. The semantic structures and interpretations are unchanged, but the denotation of an n-ary predicate must necessarily be an n-ary relation and the case sequence is, of course, irrelevant.

This section presents a notation for a class of systems called "case logics". It is reasonable to ask how the case systems of linguists, psychologists, and computer scientists fit into this framework and particularly how the distinction between surface and deep case systems is handled.

It is often argued that a language has at least two case systems, one at the surface or syntactic level and the other at a deep or conceptual level. The notation presented here is designed for both levels. Of course, at the deep level, we no longer have prepositions in the usual sense, but we do have structures which can be considered to be atomic formulas of *CL*, with variability in the order and presence of terms. Part of the understanding process is then a translation from one case system to another.

Usually a deep case system has very few cases, each one justified in terms of its presence is basic semantic constructs. A surface system may have either a profusion of cases corresponding to the many different syntactic possibilities or a small set of rather amorphous cases, such as "objective".

The translation from surface to deep cases often involves an expansion in storage structures wherein a single predicate with a few loosely defined arguments becomes a complex of more primitive predicates with arguments often missing but always well defined conceptually. It is the nature of the missing arguments which points out connections to other parts of the discourse.

In special situations this complex of primitive predicates can be simplified by elaborating the surface case *system (see* section 4). Positions *of* nodes in the semantic or conceptual network are in effect named by new surface cases whose case conditions check to see if a term could fill the appropriate conceptual position. The result is a loss in information with a corresponding improvement in efficiency for the special situation. It appears that treating such things as causation and purpose as cases (rather than as second order predicates on sentences) is such a simplification.

The next section discusses some well known case systems in terms of CL. While space limitations make the descriptions rather spare, they should indicate at least how some of the essential parts of the systems would be expressed. The problems of surface vs. deep and first order vs. second order for any particular system would need to be discussed at much greater length.

3. Some Representative Case Structure Systems

In this section we examine a variety of systems which appear suitable for description as case structure systems. For the sake of clarity, the special case C_o, which is always assumed to be in *C* is not listed in the descriptions.

3.1 Fillmore - Natural Language Cases

The most persuasive argument for the universality of cases in the deep structures of natural languages is made by Fillmore[6]. He argues from evidence in many languages that every language, regardless of its surface structures, has a deep case system which gives the relationship of many nouns to a verb.

Fillmore's description of cases and their purpose in language can be expressed directly in CL. He suggests a minimum of six cases.

Agentive (A) - the animate perceived instigator of an action
Instrumental [I) - the inanimate force or object which is causally involved.
Dative (D) - the animate being affected.
Factitive (F) - the object or being resulting from the action.
Locative (L) - location or spatial orientation.
Objective (0) - determined by verb (neutral case).

While other possible cases are mentioned we can say roughly that C = {A, I, D, F, L, 0}. Fillmore notes that the cases A and D require animate nouns, that is, for a noun to serve in the agentive or dative case it must have the feature "animate". Other features making up the set m can easily be imagined for the other cases.

Corresponding to features on nouns Fillmore says that verbs can be classified according to "sentence types" or "case frames". A case frame tells what case relationships may exist between a verb and its nouns. For example "open" may be used in four ways:

The door opened. (0)
John opened the door. (A 0)
The wind opened the door. (I 0)
John opened the door with a chisel, (A 0 I)

We can represent the case frame for "open" as [0 [A) (I)]. This says that when "open" is used its object must appear but the agent and instrument are optional. Clearly a case frame is a case structure *(£)* in CL.

Fillmore shows by example the case *signals* (S.) of various languages. He also gives some tentative rules for English. For example:

"The A preposition is by; the I preposition is by if there is no A, otherwise it is with; the 0 and F prepositions are typically zero; the B [benefactive case] preposition is for; the D preposition is typically to....

"If there is an A it becomes the subject; otherwise, if there is an I, it becomes the subject; otherwise the subject is the 0."

3.2 Simmons -¡ Semantic Networks

R.F. Simmons[20] gives an extensive account of semantic networks and their use in processing natural language by computer. An essential feature of these networks is a set of deep case relations connecting nominal concepts (underlying noun phrases) to verbs. While a semantic network does not require any specific case structure Simmons has chosen one which follows from work of Celce-Murcia and Fillmore".

In his system there are five deep case relations, so that *C* = (causal actant, thome, locus, source, and goal}. The set of prepositions, or *Q.,* is of course just the prepositions of English. Verbs are classified into paradigms according to the case sequences they allow, a paradigm, then, is a case structure (3). For example, the ergative paradigm consists of the sequences (for the active voice):

(causal-actant., theme, causal-actant)
(causal actant., theme)
(causal actant-, theme)
(theme)

Simmons gives "break" as an example of an ergative verb. Thus

John broke the window with a hammer.
John broke the window
The hammer broke the window
The window broke

are all well-formed since in each case one of the case sequences is matched (where "John" is the causal-actant., "window" is the theme, and "hammer" is the causal-actant).

Another example is the "reflexive-deletion paradigm where the theme is deleted if it corresponds to the CA1 [causal-actant.]". Thus "run" may be used in several ways:

John ran to school
John ran a machine
The machine ran
The brook ran

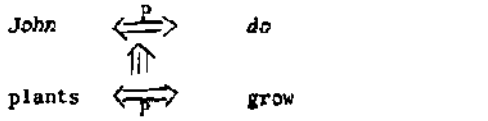
In each of the sentences there is a theme - John, machine, or brook. The paradigm allows the deletion of the theme if it is the same as the causal-actant. In this the case structure is

(causal-actant, goal)
(causal-actant, theme)
(theme)

Simmons discusses other aspects of the case structure portion of semantic networks as well. He points out that a "semantic definition" of a verb can be given by describing the properties of the nominal concepts serving in each case relationship to the verb. It appears to be the case that these descriptions can be adequately handled by means of our case conditions. If so then a parser for semantic networks would simply check the descriptions in the process of determining well-formedness.

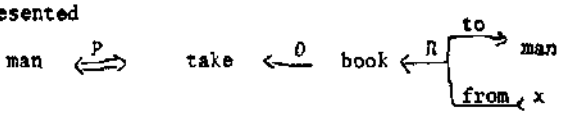
3.3 Schank - Conceptual Parsers

Roger Schank[17] has developed a theory of discourse comprehension based on dependency grammar. In his system "conceptualizations" are the primitive objects underlying natural language utterances. These conceptualizations are inferred from the discourse as a whole in such a way that the syntactic structure of an individual sentence may bear only a secondary relationship to its conceptualization. For example the structure underlying "John grew plants" is



where "P" means "past", " $\Longleftrightarrow$ " means the actor-action relation and "$\Uparrow$" means causation. A "conceptual parser" would have to know that John did an action and that that action caused the plants to grow.

Schank says that an action can have any of four "dependents" - object, recipient, direction, or instrument. The sentence "The man took the book" is represented



which indicated that the man is the recipient (R) of the object (O) book from someone (X). More generally we can think of a verb "trans" with the structure

"give" and "receive" are realized when z=x and "take" when z=y.

It seems clear that, for Schank, $\mathcal{C}$ = {actor, object, recipient, direction, instrument} where the actor in an actor-action pair is considered to be a case. Schank has a large set, $\mathcal{M}$, of features which are checked by the case conditions R_{C_i}. The case structures, $\mathcal{D}_p$, of predicates (verbs) in Schank's system are sets of sequences composed of the five cases. The extensive checking of features coupled with the fact that the conceptual network marks holes for missing cases reduces the need for tight restrictions on the $\mathcal{D}_p$'s.

3.4 Kulikowski - Medical Modeling

Another system which appears unlike the others and yet fits naturally into CL is the glaucoma model of Kulikowski and Weiss[11]. In their model various primitive descriptors of physiological conditions or states (e.g. "pressure", "atrophy", "age", "adhesion", etc.) are given attributes (e.g. "medium:, "location", "time", "agnitude", etc.). The specification of a state is an assignment of values to some or all of the various attributes for a descriptor.

Describing the medical model in CL, we would say that the primitive descriptors are predicates (elements of $\mathcal{P}$) and the attributes are cases (elements of $\mathcal{C}$). The restrictions on the objects which may fill attribute slots are the features (elements of $\mathcal{M}$) of CL. The case structure ($\mathcal{D}_p$) of a primitive descriptor, P, is the set of rules which determine whether of not an attribute may be omitted. For example, it seems necessary to state the location of pressure in every case but the medium may be unspecified.

It is interesting to note that the medical model introduces time and location, typically adverbial constituents, as attributes of descriptors. Fillmore[6] suggests a similar collapsing of (many) adverbs to cases in English. In fact, a commitment to cases often leads to a reduction in the number of grammatical categories - verbs, adjectives and some nouns become predicates, and adverbs and other nouns fill the case positions.

The fact that the Kulikowski-Weiss model can be stated in CL is significant in two ways. First, it suggests that for convenient, efficient, or flexible implementation of such a model a computer program should be amenable to case structure systems. Second, by providing a connection to natural language it indicates a direction to follow when adding a natural language interface to the model.

4. CHRONOS II - A Computer Program for Understanding Natural Language with a Flexible Case Structure

CHRONOS II is a natural language processor written in LISP 1.6 for a PDP-10 computer. The name and a small portion of the program are derived from an earlier work which emphasized the time relations in language (Bruce[2]). The program is discussed more fully in Bruce and Singer[3]. Here we will mainly look at the case structure mechanisms.

The storage structures in CHRONOS II are essentially simple sentences (events, atomic formulas) connected by logical connectives, causation links, and temporal relations. Each simple sentence has a verb (predicate) and a number of cases. These cases may include important adverbial modifiers and temporal indicators as well as the usual nominal cases.

The parser is basically an augmented transition network (Woods[22], Kaplan[9]). Important states appear as separate LISP functions. The sentence generator is currently just a mapping from storage structures to stereotyped sentences, but in gross form it mirrors the parser. Deduction in CHRONOS II is also limited to specific areas which have been of interest in applications, such as time, causation, and certain summarizations. However, since the storage structures retain most of the information in a sentence in a form easily transformed to predicate calculus there is no reason why more general deductive powers could not be added.

The features ($\mathcal{M}$) on nouns and case structures ($\mathcal{D}_p$) of predicates are given by the lexicon. A noun entry includes the following information:

[word]
word-class --- N
properties --- P_1, P_2,...P_n

P_1 --- [allowed values for P_1]

P_2 --- [allowed values for P_2]

:

P_n --- [allowed values for P_n]

supersets --- N_1, N_2,..., N_j

subsets ---N_1, N_2,..., N_k

For example the noun human might have a property "state-of-mind" with allowed values being "happy", "sad", or "blue". It could have the superset "animal" and the subset "woman", Naturally any noun will have the properties of its supersets in addition to those explicitly listed. A verb entry has the form:

[word]

word-class ---V

paradigm --- (C_1, C_2,..., C_n)

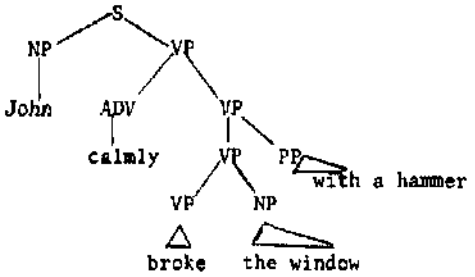
The paradigm for a verb is an ordered set of cases. In a given simple sentence CHRONOS II will (currently) allow any or all of the cases to be missing. Thus the case structure for a verb is a portion of the power set of paradigm subject to restrictions on features.

Case determination depends on syntactical information (case signals) as well as feature checking (case conditions). Currently in CHRONOS II the definitions of case signals and case conditions are combined in one LISP function for each case. Someone who wishes to use CHRONOS II for a specific application must specify the list of cases, define each case, and modify the lexicon appropriately.

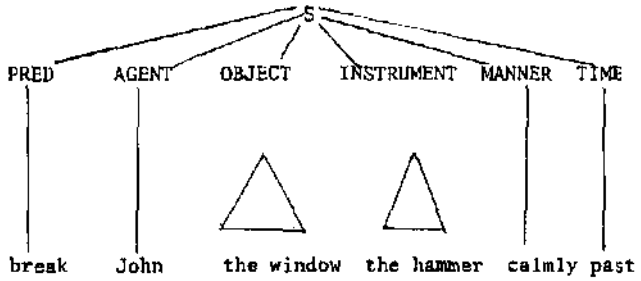
Each possible case function is called for every preposition term (noun phrase, prepositional phrase, or adverb) and returns a number giving the likelihood that the given preposition term serves that case relationship to the main verb of the sentence. The case whose function returns the highest value is temporarily assigned to the preposition term. A failure, consisting of values of D for all the cases, forces a backup to the previous term and a reassignment of *the* case for that term.

The case parser[21] is essentially independent of the particular set of cases being used. Its speed and generality are of course directly dependent on the accuracy of the likelihood numbers returned by the case functions. There are various features which extend the simple heuristic value assignments, making the case functions easier to write or the parsing more efficient. For example, a variable number of "pre-emptive levels" can be created. Once a case function returns a value at a new and higher pre-emptive level then only values at that level (or higher) are considered valid. This is especially useful when one knows that a particular preposition, say, signals one and only one case. Then that case function returns a value which pre-empts any previous use of that case.

CHRONOS II has been used primarily with two case systems. One is for the analysis of medical records, i.e. nurse's notes, and is the more general of the two. In it c = {agentive, objective, dative, locative, instrumental, temporal, frequency, derivative, manner) where agentive, objective, dative, instrumental, and locative are as in Fillmore[10], temporal points to the duration of an event, frequency is indicated by adverbs such as "rarely" and "often", derivative is indicated by words such as "increasing", and manner includes most other adverbs. The effect of such a case system can best be seen by comparison with a traditional C = {subject, object, indirect object} system. For example, the parse of "John calmly broke the window with a hammer" might, in the traditional system, be

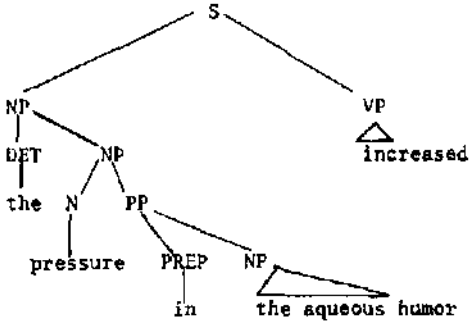


where the triangles (Δ) indicate possible fine structure of no importance to the example. Note that the traditional subject-predicate organization is maintained resulting in a rather opaque representation. In the system used by CHRONOS II the above sentence would be represented

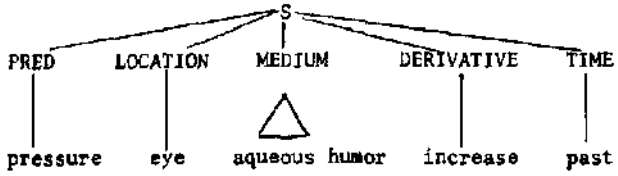


The latter structure is not only easier to understand but is also more efficient for the program.

The other case structure used in CHRONOS II is needed when CHRONOS II serves as a model for the eye disease, glaucoma (see Kulikowski-Weiss[11]). In it $\mathcal{C}$ = {agentive, objective, locative, instrumental, temporal, frequency, derivative, medium} where medium indicates a fluid location. This allows a distinction between "in the blood" and "in the blood vessel". We are considering adding other cases as the need for finer distinctions arises. An example for this case system is the sentence "the pressure in the aqueous humor increased". Again, a traditional parse



seems less useful than one which arises from cases suited to the problem.



Note that in the second parse, "pressure" has become the predicate. It is an essential principle of CHRONOS II that the predicate of a simple sentence is the logical predicate (relative to the purpose of discussion) rather than the syntactical predicate (usually the verb).

5. Discussion

This report presents a notational system (CL) for case structure systems and examines a few existing systems in terms of the notation. CL is *a* first order system which appears useful in characterizing case systems and will hopefully lead to a better understanding of natural language storage structures.

Several observations about case structure systems follow from this study, observations which are actually open problems. For example, consider the sentence "John gave birth to a baby". The oddness of this sentence can be handled in at least three ways. First, we could say that "to give birth" is a predicate which imposes the case condition on the agent that it be female. In that case the sentence is not a formula of CL. CHRONOS II could do that, or alternatively, raise its eyebrows by accepting the sentence at a lower pre-emptive level (see section 4). Second, we could say that "to give" is a predicate, that the sentence is well-formed, but its negation is provable from a set of usually accepted facts about life. Third, we could say that "to give" is the predicate but demand that the sentence be treated as ill-formed. The last approach is feasible as long as the conditions on well-formedness can be stated as predicates on the single sentence (such as features) but is probably risky if it requires examination of other formulas.

An interesting problem, proposed in section *2,* concerns the interaction of case signals and case conditions. In CHRONOS II these functions mesh in a rather haphazard way. It should be interesting to examine situations where priorities in their use affect efficiency.

Case systems, since they emphasize a logical structure, rather than a purely syntactic one, lessen the importance of the syntactic predicate and of the sentence as a unit. Schank[17] and others have stressed the need to examine a discourse as an integrated whole rather as a collection of isolated sentences. It seems that the better a case system is, that is, the more relevant it is to the problem solving situation at hand, the easier it is to connect sentences in the discourse in meaningful ways.

Since many case systems are in use, a natural question arises, "What is the best case system?" My experience with CHRONOS II suggests that, at this stage in the development of intelligent programs, one can only speak of the goodness of a case system relative to a problem situation. The finely distinguished cases which are arising in the medical model version of CHRONOS II would probably only clutter the program which analyzes nurse's notes. Conversely, certain cases which one would need in discussing everyday life would be unnecessary in a medical model. An important problem then is to decide what cases to use in a particular application. Even more interesting might be a study of transformations between various case representations. Problems of summarization and analogical reasoning will probably be more tractable with a better understanding of case structure transformations.

Acknowledgements

Many people have contributed to this work. Discussions with Robert Simmons first interested me in case logics. William Fabens, Richard Orgass, and Ann Yasuhara made useful suggestions on case logic, Neil Singer, Santosh Chokhani, Nanette Yaskin, and Mike Trigoboff all worked on CHRONOS II and contributed to discussions on case systems. I also want to thank the referees for their comments and suggestions which I have incorporated.

References

1. Bobrow, Daniel. "Natural language input for a computer problem solving system". In Semantic Information Processing (Ed: Marvin Minsky) MIT Press, 1968.

2. Bruce, Bertram. "A model for temporal references and its application in a question answering program". Artificial Intelligence 3, 1972 pp. 1-26.

3. Bruce, Bertram, and Neil Singer, "CHRONOS II: the processing of incompletely specified data". Computer Science Department, Rutgers-NIH report CBM-TR-9, 1972.

4. Celce-Murcia, M. English Comparatives. Ph.D. Thesis, UCLA, 1972.

5. Chomsky, Noam. Aspects of the Theory of Syntax. MIT Press, 1964.

6. Fillmore, Charles. "The case for case". In Universals in Linguistic Theory (Ed; Bach and Harms), Holt, Rinehart, and Winston, 1968.

7. Green, Cordell, and B. Raphael. "Research on intelligent question answering systems". Proc. ACM 23rd Nat. Conf., 1968.

8. Green, B.F., A.K. Wolf, Carol Chomsky, & Kenneth Laughery. "Baseball: An Automatic Question Answerer". In Computers and Thought (Ed: Feigenbaum and Feldman), McGraw-Hill, 1963.

9. Kaplan, R.M. "Augmented transition networks as psychological models of sentence comprehension". Artificial Intelligence 3, 1972. pp. 77-100.

10. Katz, J.J. The Philosophy of Language. 1965.

11. Kulikowski, C. and S. Weiss. "Computer based models for glaucoma". Computer Science Department, Rutgers-NIH report CBM-TR-3, 1971.

12. Lindsay, Robert K. "Inferential memory as the basis of machines which understand language". In Computers and Thought (Ed: Feigenbaum and Feldman), McGraw-Hill 1963.

13. Martin, William A., Rand B. Krumland, and Alex. Sunguroff. "More MAPL: Specifications and basic structures". Automatic Programming Group, Project Mac, MIT, Internal Memo 8, 1973.

14. Norman, Donald A. and David E. Ruraelhart. "Active semantic networks as a model of human memory". Third International Joint Conf. on Artificial Intelligence, 1973

15. Quillian, Ross. "The teachable language comprehender: a simulation program and theory of language". Comm. ACM 12, 1969, pp. 459-476.

16. Sandewall, Erik. "A programming tool for management of a predicate-calculus-oriented data base". Second Int'l .Joint..Conf\. on Artificial Intelligence, 1971.

17. Schank,. Roger. "Finding the conceptual content and intention in an utterance in natural language conversation". Second Jnt'l Joint Conf. on Artificial Intelligence, 1971.

18. Schwarcz, R.M., J.F, Burger, 6 R.F. Simmons. "A deductive question answerer for natural language inference". Comm. ACM 13, 1970, pp. 167-163.

19. Shapiro, Stuart. "A net structure for semantic information storage, deduction and retrieval". Second Int'1 Joint *Conf.* on Artificial Intelligence, 1971.

20. Simmons, R.F. "Semantic networks: their computation and use for understanding English sentences". Computer Science Department University of Texas at Austin-CAI Lab report NL-6, 1972.

21 Trigoboff, Mike. "A heuristic case structure parser". Computer Science Department, Rutgers-NIH report CBM-TM-16, 1973.

22. Woods, W.A. "Transition network grammars for natural language analysis". Comm. ACM 13, 1970, pp. 591-602.