

AN OVERVIEW OF
AN AUTOMATICALLY COMPILABLE HIERARCHICAL DEFINITION MATCHER

Barbara C. Sangster*
Laboratory for Computer Science Research
Rutgers University
New Brunswick, N.J. 08903

The problem addressed in this paper is that of determining whether the conditions expressed in the definition of a concept are satisfied by a set of assertions (a task of attempting to match the assertions to the definition). The approach presented here is to precede the matching activity proper with a one-time preprocessing phase, during which the definition is automatically transformed from a rather complex form in which it is originally expressed into a simpler form which appears to be more suitable to the matching task at hand. Use of this transformation process permits the functions of recognition and interpretation to be separated, thereby improving the efficiency of the matching process. This separation is performed by distinguishing those components of the definition which must be found to be true from those whose truth may subsequently be inferred and those whose truth is irrelevant to the matching process. The transformation is performed by means of a process of symbolic instantiation of the definition -- the translation of the definition from a set of criteria for satisfying the definition into an exemplary instance of the concept itself. The algorithm for the transformation and the subsequent matching process are outlined and illustrated in the text.

1. FORMULATION OF THE TASK

This paper describes an approach to a problem that has arisen within the TAXMAN II project on Legal Reasoning [4], [5], [6], [7], [9] -- the design and implementation of a MATCHER that will correctly identify exactly those subcases within a case description (i.e., that will identify those subsets of the facts of a legal case) that satisfy the conditions expressed in the definition of some legal concept. As the purpose of the TAXMAN II project is to

investigate the structure of legal concepts, and of decisions based on those concepts, through the representation of those concepts and decisions in AIMDS, it is evident that a MATCHER will be required to play a central role in the envisioned system. The MATCHER described in this paper was designed to meet that need.

Work in the first few months of the TAXMAN II project has centered on a single concept in the domain of corporate tax law -- a kind of corporate reorganization called a BREORGANIZATION. A BREORGANIZATION is a complex concept, involving the exchange between two corporations (related in certain ways) of property bearing a particular, rather intricate, relationship to those two corporations. The initial task posed for the MATCHER (the task with which we shall be concerned in this paper) was, therefore, to take a case description consisting of a possibly very large set of possibly very intricately related facts, and to identify every subcase (i.e., every subset of those facts) that indicated the existence of some instance of a BREORGANIZATION.

A later version of the MATCHER is expected to be able to take a set of legal concepts

*The research reported in this paper was partially supported by the National Science Foundation under Grant #SOC-7811408 and by the Research Foundation of the State University of New York under Grant #150-2197-A.

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the National Science Foundation or the Foundation of the State University of New York.

(AREORGANIZATION, BREORGANIZATION, CREORGANIZATION, and so forth), and, in a single pass, identify for each concept all those subcases satisfying its definitional conditions. Another extension to the MATCHER that we contemplate is of a different kind: the current MATCHER identifies a subcase S as a match to the definition if and only if S exactly meets the definitional conditions; a future version of the MATCHER will permit, in the absence of an exact match, a sequence of gradual deformations until a match to S has been found.

Let us return to the version of the MATCHER currently under development. Careful examination of the definition given for BREORGANIZATION and of representative case descriptions disclosed

1) that some of the assertions comprising the definition were irrelevant to the determination of whether the definitional conditions had been met -- that is, they did not need to be true; these clauses seemed somewhat better suited to a description than to a definition;

2) that some other assertions comprising the definition, while they did have to be true, could be inferred to be true on the basis of certain other knowledge; the issue of mutual inferrability was not directly addressed, as mutually inferrable sets of assertions in this domain appeared to be expressible in terms of a canonical set of clauses whose truth had to be observed -- that is, clauses of neither type 1 nor type 2.

For example, consider that the following is part of a definition of a ZOPPLE

Every creature with a beak and feathers is a ZOPPLE.

Every ZOPPLE has wings.

A ZOPPLE may have orange feathers.

If we encounter a creature with a beak and feathers, then we know it is a ZOPPLE without concerning ourselves further. The color of the feathers (type 1 above) is irrelevant to our classification of the creature; the presence of wings may be inferred (type 2), once the creature has been identified.

From the characterization of the problem given above, we may readily admit the desirability of

SPEC1) reducing combinatoric problems wherever possible,

SPEC2) identifying as a valid BREORGANIZATION every subcase for which all of the conditions included within the definition -- aside from assertions of the types described in 1 and 2 above -- were true, and subsequently devising some means of filling in whatever additional information could then be filled in.

In terms of the example presented earlier, we might identify as a ZOPPLE any creature with a beak and feathers, subsequently inferring the presence of wings, and adding the fact the feathers are orange, wherever this is true of the ZOPPLE in question.

In view of our plans for later phases of the work, we should also like our design to permit the easy representation of the process of partial matching.

SPEC2 permits us to characterize our task in terms of the paradigms of recognition and interpretation. Recognition is the process of determining whether or not a particular object satisfies any of a set of conditions -- i.e., given an object and a set of sets C of conditions, it returns a yes or a no for each C. Interpretation associates with an object not only a classification -- or a set of classifications (recognition), but (for at least one of the classes) additional material as well, such as a structural description. For example, a structural matcher, by distinguishing between the presence and absence of an isomorphism between two networks, performs recognition; a parser, by distinguishing between the grammatically valid and invalid strings presented to it (recognition), and associating at least one parse tree with every grammatically valid string, performs interpretation. Note that a parser may be viewed as performing recognition by means of interpretation.

As we can see, our task may be decomposed into two modules -- one in which the recognition of valid BREORGANIZATIONS is performed, and one in which an interpretation is produced for every valid BREORGANIZATION. Note that, by this method, the classification of a case as a valid or invalid instance of a BREORGANIZATION would be completed by the recognition module, and that, unlike the parsing analogue, our interpretation module would be required only for the creation of the appropriate structural descriptions.

This design strategy

1) would successfully avoid certain combinatoric problems often encountered by parsers

a) by eliminating the requirement that the conditions for recognition failure (i.e., the conditions requiring a no to be produced by the recognition module) must be identified through the failure of interpretation (i.e., through the inability of the interpretation module to create an interpretation), and

b) by removing the necessity of generating interpretations for matching subcases when only recognition was desired,

2) would successfully avoid a problem encountered by many structural matchers, by eliminating the possibility of recognition failure caused solely by the absence in the case description of non-essential information which was present in the original definition, and

3) would permit the computation of missing but inferrable information to be performed by the interpretation module.

Before we proceed to a discussion of the means selected to meet the two specifications given above, let us look more closely at the way in which the definitions and the case descriptions are represented. The representation is in AIMDS, a frame-based language [8]. AIMDS allows the representation of a set of types, a set of instances of the types, a set of typed relations that may be asserted to hold between pairs of instances of the appropriate types, and a set of pattern-directed inference rules called consistency conditions (CCs) which specify the allowable combinations of assertions about relations between instances.

One of the attractions of representing sets of assertions in AIMDS is that those representations have an analogue in network form. Several powerful methods have been devised for manipulating structures which permit a network representation. Two such methods involve the use of recognition networks and structural matchers for networks.

A recognition network performs much like a parser for pieces of subnet, performing recognition by assigning a full semantic net to the input subnets (interpretation). An automatically compilable recognition network

(ACORN), developed by F. Hayes-Roth and D. J. Mostow [3], has been used in the understanding of connected speech. A structural matcher, on the other hand, performs recognition by determining whether every component of a criterial network has a unique counterpart in the network whose recognition is being attempted. A program with the capability of structural matching, called DMATCH, was developed by D. Touretzky [12] for use with AIMDS.

In order to see to what extent either or both of these methods might be applicable to our own task, let us look more closely at the degree to which the representations in AIMDS of definitions and case descriptions have analogues as networks.

A case description is a set of assertions about relations between instances. For example, an assertion in a legal case description represented in AIMDS might say that the instance TRANS-1 of the type TRANS has in AGENT relation the instance ACTOR-1 of the type ACTOR. A case description may be represented as a network in which the nodes correspond to the instances, and the links correspond to the relations between them.

A definition, which specifies all structures meeting a set of conditions, consists of 1) a set of propositions specifying the allowable relations between instances, and 2) a set of consistency conditions (CCs) expressing constraints on how the propositions in the first set may be instantiated. For example, in the domain of corporate tax law, propositions of the first set might state that an instance of the type TRANS (a transfer of property) may have an instance of ACTOR in AGENT relation, and may have an instance of ACTOR in OLDDOWNER relation. A CC in this domain might state that an instance of TRANS may have an instance of ACTOR in AGENT relation if and only if that instance of ACTOR is also in OLDDOWNER relation to that instance of TRANS. The propositions of the first set may be represented as a network in which the links correspond to relations, and the nodes stand, not for instances, but for the types of instances. No obvious analogous way of also representing CCs in network form presents itself. (Possible graphical representations of CCs, such as those based on Petri nets, would not appear to have useful analogues in case descriptions.)

These observations have failed to show us how we might formulate an analogue to a definition in solely network form (as we have said that an analogue to a case description may be

- 1) We have seen that only a subset of the assertions in the definition need be found to be true in a subcase S for S to match the definition.
- 2) We have seen that, given two networks (N1 and N2), it is possible to use a structural matcher to recognize that every component of N1 has a unique counterpart in N2.
- 3) We have seen that methods have been devised for generating a full semantic net (an interpretation) from a set of subnets.

the original ACORN to fill in whatever missing assertions we can (interpretation).

Such a strategy, if it could be implemented, would have clear advantages. By permitting the representation of the definitional conditions in a single mode, it would be conducive to easy understanding and, most probably, economy of storage. By permitting recognition to take place through a single kind of process, it would be likely to further enhance comprehensibility and storage economy, as well as efficiency of operation. For example, elimination of the need during recognition for pattern-directed inference processes (which very likely would have been used if interpretation were also being done at this time) would permit recognition to be performed in LISP directly on the data structures without the overhead required for running AIMDS (which is written in the language FUZZY, which is itself written in LISP). Furthermore, since the single representational mode (that of a network) and the single recognition process (that of structural matching) have been the objects of both theoretical and empirical study by many researchers, the strategy outlined above would most probably enable us to draw on the results of others in improving our own design and implementation.

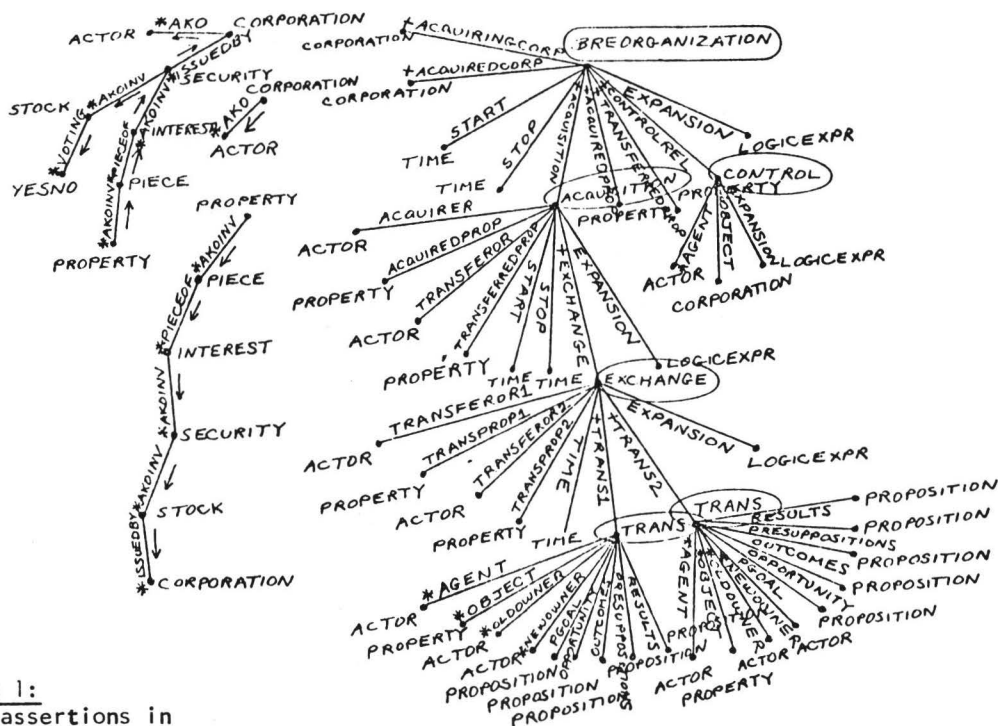


Figure 1:
The assertions in
the definition.

QUESTION2) can we generate an ACORN module to fill in all the information that is missing in the case description but can be inferred from other knowledge, and if so, how?

2. SKETCH OF A SOLUTION

An answer to the questions posed above was devised, and is outlined below.

to take all the assertions (in the form of relations between typed variables) present in the definition, and, for all the variables contained in the assertions, to perform, using the consistency conditions (CCs), a process of symbolic instantiation of the variables, in such a manner that two variables are bound to the same instance if

and only if they are required to have the same binding in all case descriptions satisfying the definition. (The use of a single symbolically instantiated form is sufficient for the syntactic class of definitions under discussion here. These definitions contain conjunction, but neither disjunction nor negation, either of which would require a more complex symbolically instantiated structure. For further discussion, see [6] and [7].) The assertions in the definition for a BREORGANIZATION (before symbolic instantiation) are illustrated in Fig. 1.

We can then divide the assertions (now in the form of relations between instances) into four sets:

A) those whose truth must be observed in the case description -- these will provide us with all the bindings we are required to identify in S (The assertions of set A for a BREORGANIZATION are illustrated in Fig. 2.);

B) those which can be used to pass these bindings on as the structural description is built up during interpretation;

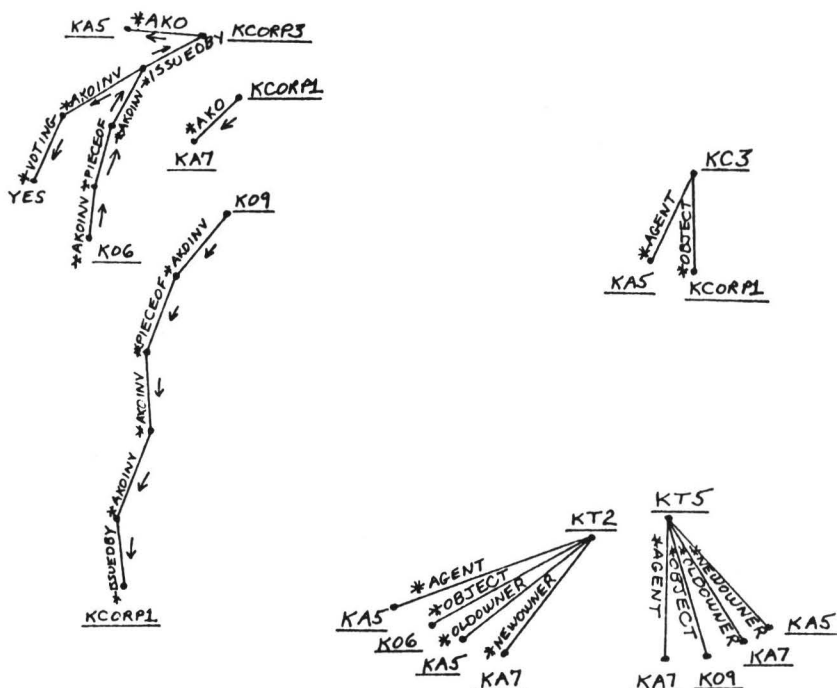


Figure 2:
The assertions of set A.

C) those whose truth can be inferred, on the basis of the CCs, from that of the assertions in sets A and B;

D) those whose truth is not relevant to a determination of whether the definition has been satisfied.

Sets A, B, and D, each of which now consists of sets of instantiated assertions, can then be used to produce the recognition and interpretation modules characterized above. Set A is used to create the network to which a structural match of a subcase can be attempted by the recognition module (N1 in the preceding section). We call this the KERNEL structure for the definition (or, more simply, the KERNEL structure). The assertions of set B are transformed into a set of commands to make additional assertions -- that is, to add new assertions to those already present -- this plays part of the role of the ACORN module mentioned in the preceding section, and creates the first part of the desired interpretation. We call this set of commands the INNER ACORN generator. The assertions of set C are discarded. After the first part of the interpretation of a recognized subcase S has been completed, those assertions of set D that are true for S are used to add (with additional commands) to the structural description of S, thereby completing the role of the ACORN module, and concluding the interpretation process. This

additional set of commands, which we call the AUGMENTED ACORN generator, augments the interpretation of the recognized subcase with clauses of a descriptive, rather than a definitional, character. We call the combination of the INNER ACORN generator and the AUGMENTED ACORN generator simply the ACORN generator.

The compilation process sketched above thus results in a KERNEL structure and an ACORN generator, which can then be used for recognition and interpretation whenever a case description is presented for comparison with the definition. The compilation and execution processes are illustrated in Figures 3, 4, and 5.

The way in which the compilation algorithm has been formulated makes it possible to construct an informal proof that the algorithm correctly transforms any well-formed definition with the purely conjunctive structure sketched above. The algorithm and the proof are presented in detail in [6]. Details on representational issues involved in our method for manipulating definitions may be found in [7].

3. EXECUTION

In the preceding section, we saw how the compilation of the KERNEL and ACORN for a definition with a conjunctive structure

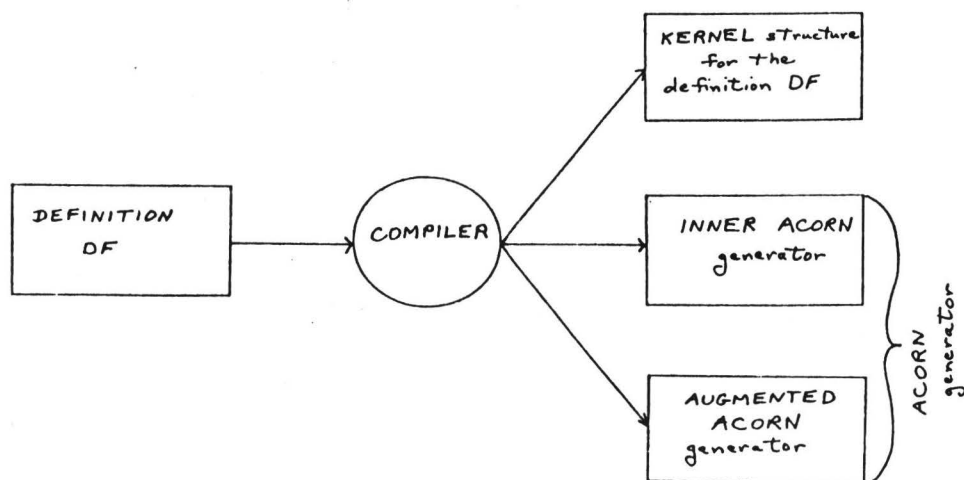


Figure 3:
Compilation of KERNEL structure and ACORN generator.

proceeds. Once this compilation has been completed, it need never be repeated (unless the definition itself should change), and the compiled KERNEL structure and ACORN generator may be used directly for comparison with the definition whenever a case description is presented.

The matching of subcases against the KERNEL structure takes place first; at the option of the user, this process may terminate with the identification either of a single matching subcase, or of a larger set of matches (including the set of all matching subcases). When the desired number of matching subcases have been recognized (or when all cases in the case description have been examined), control returns to the user's program. At this point, any procedures of the user's choice may be executed, including (whenever desired) execution of the ACORN generator to produce interpretations for any or all of the matching subcases identified by the recognition routine.

In order to control possible combinatoric difficulties, the KERNEL is decomposed into a set of small networks, against each of which all substructures of the same type in the case description are tested for an exact DMATCH (STAGE1). (We hope to be able to select "small networks" from the KERNEL in such a way that matching to any single small network will involve a minimal degree of combinatoric complexity.) The substructures that survive STAGE1 (and no others) are then combined in all possible valid ways into larger networks of some degree of increase in complexity. A DMATCH of each of these structures with the corresponding substructure of the KERNEL is then attempted, and bindings constraints between formerly

separate components of the new network are thereby tested. This process is repeated with surviving substructures until the DMATCH is conducted against the KERNEL structure itself. This final stage of DMATCHing will be terminated as soon as the desired number of matching subcases have been recognized. If this stage is permitted to run to completion, the survivors represent all and only the subcases in the case description that meet the conditions expressed in the definition.

The execution of the recognition component is illustrated in Fig. 6. For this example, 5 instances of TRANS (T1, T2, T3, T4, T5), 2 instances of CONTROL (C1, C2), and 2 instances of OBJECT (O6, O9) were used. The value of MAKEFULLLIST shows the survivors of STAGE1. The value of BGO shows the single valid instance of a BREORGANIZATION that can be created from these components.

If interpretations of any of the identified matches are desired, the ACORN generator may then be used to produce them.

4. IMPLEMENTATION

Implementation activity on the matcher has taken two directions. First we constructed and ran an automated demonstration of the execution of a hand-compiled KERNEL and ACORN. Having demonstrated the feasibility of the design for our application, we began work on the compiler for the KERNEL and ACORN, concurrently refining the algorithm outlined in this paper, and presented in detail in [6]. This effort has so far yielded encouraging results.

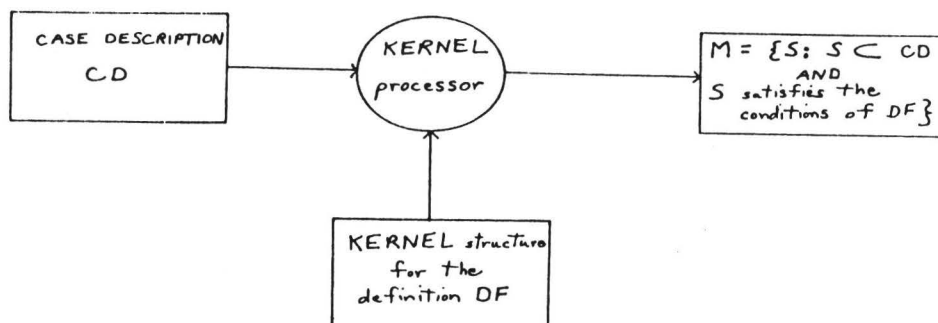


Figure 4:
Execution of recognition component,
using the compiled KERNEL structure.

5. IMPLICATIONS

The primary attractions of the definition matcher appear to stem from the separation of recognition and interpretation, and from the reduction of the recognition process to the structural matching of networks. This design strategy significantly enhances matching efficiency by eliminating the requirement that the absence of a match must be recognized through the failure of interpretation, by removing the necessity of generating interpretations for matching subcases when only recognition is desired, and by permitting the recognition process to be performed without the overhead associated with the use of pattern-directed inference rules. In addition, this strategy eliminates the possibility of match failure caused solely by the absence of non-essential information, and permits the computation of missing but inferrable information to take place during the interpretation phase.

In addition to its immediate practical advantages, the use of the KERNEL+ACORN paradigm seems not only to induce a more accurate representation of definitions, but also to provide a sound basis for the expected development within the Legal Reasoning Project of a deformation-oriented definition matcher.

ACKNOWLEDGEMENTS

The author thanks S. Amarel, V. Ciesielski, L. T. McCarty, T. Mitchell, N. S. Sridharan, D. Touretzky, and an anonymous referee for providing some very helpful suggestions and comments on an earlier draft of the paper, and especially for identifying areas of particular obscurity. The author is also very grateful to N. S. Sridharan and D. Touretzky for speedy assistance in adding a new capability to AIMDS required in an early phase of the implementation effort. It should, however, not be inferred that any of the above necessarily endorses all the conclusions reached in this paper -- with the sole exception of the author, who takes full responsibility for its contents.

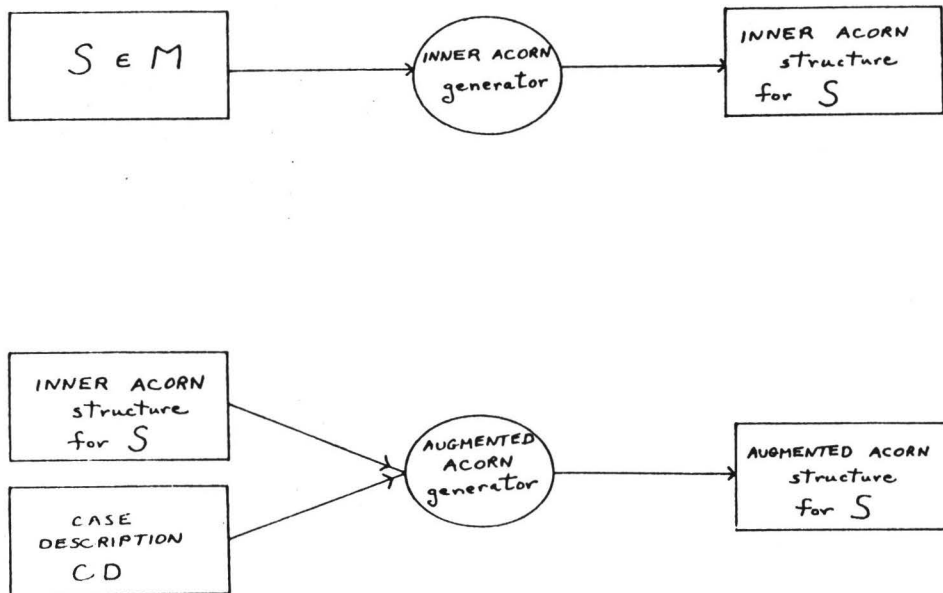


Figure 5:

Execution of the interpretation component, using the compiled INNER ACORN generator and AUGMENTED ACORN generator.

REFERENCES

- [1] Freuder, E. C. 1978. "Synthesizing Constraint Expressions". CACM, vol. 21, pp. 958-966.
- [2] Haralick, R. M. and L. G. Shapiro. 1979. "The Consistent Labelling Problem: Part I". IEEE Transactions on PAMI, vol. 1, pp. 173-184.
- [3] Hayes-Roth, F. and D. J. Mostow. 1975. "An Automatically Compilable Recognition Network for Structured Patterns". Proceedings of IJCAI-75, vol. 1, pp. 246-251.
- [4] McCarty, L. T. 1977. "Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning". Harvard Law Review, vol. 90, pp. 837-893.
- [5] McCarty, L. T., N. S. Sridharan, and B. C. Sangster. 1979. "The Implementation of TAXMAN II: An Experiment in Artificial Intelligence and Legal Reasoning". Rutgers University Report #LCSR-TR-3.
- [6] Sangster, B. C. 1979a. "An Automatically Compilable Hierarchical Definition Matcher". Rutgers University Report #LRP-TR-3.
- [7] Sangster, B. C. 1979b. "On the Automatic Transformation of Class Membership Criteria". Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics. Univ. of California at San Diego, La Jolla, CA.
- [8] Sridharan, N. S. (ed.) 1978a. "AIMDS User Manual, Version 2." Rutgers University Report #CBM-TR-89.
- [9] Sridharan, N. S. 1978b. "Some Relationships between BELIEVER and TAXMAN". Rutgers University Report #LCSR-TR-2.
- [10] Sridharan, N. S. and C. F. Schmidt. 1978. "Knowledge-Directed Inference in Believer". In Pattern-Directed Inference Systems, ed. by D. Waterman and F. Hayes-Roth.
- [11] Srinivasan, C. V. 1976. "The Architecture of Coherent Information System: A General Problem Solving System". IEEE Transactions on Computers, vol. 25, pp. 390-402.
- [12] Touretzky, D. 1978. "Learning from Examples in a Frame-Based System". Rutgers University Report #CBM-TR-87.
- [13] Woods, W. A. 1975. "What's in a Link: Foundations for Semantic Networks". In Representation and Understanding, ed. by D. G. Bobrow and A. Collins. Academic Press.

SECOND-CONTEXT >> (BGO)

Enter MAKEFULLLIST:

```
! PROTS = (PROTOTRANS1 PROTOTRANS2 PROTOCONTROL1 PROTOO9 PROTOO6)
MAKEFULLLIST = ((O6) (O6 O9) (C1 C2) (T2 T4 T5) (T2 T4 T5))
```

((T2 T5 C2 O9 O6) NIL)

Figure 6:
Sample execution of the recognition component.