

REPRESENTATION AND EFFICIENCY IN A PRODUCTION SYSTEM FOR SPEECH UNDERSTANDING

Donald L. McCracken
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

ABSTRACT. The research question studied is whether it is practical to use a production system (PS) architecture for a version of the Hearsay-II (HSII) speech understanding system. The study focuses on adequacy of representation, space efficiency and time efficiency. A running PS architecture called HSP was designed and implemented and twelve HSII knowledge sources were translated to HSP productions, with two of the twelve actually run under HSP. Detailed comparisons of HSII and HSP produced the following results: (1) HSP is adequate to represent the HSII speech knowledge, even though HSP is a comparatively simple PS architecture; (2) HSP suffers a moderate space penalty for representing declarative HSII knowledge, which is serious since HSII contains a high proportion of such knowledge; (3) representation of HSII declarative knowledge as HSP productions causes problems with multiple use of that knowledge; (4) lack of a local working memory in the HSP architecture has serious consequences both for space and time efficiency; and (5) HSP has a time efficiency handicap of two to three-and-a-half orders of magnitude, in spite of efficiency mechanisms which make HSP comparable in time efficiency to other PSs. The paper ends with a set of questions that remain to be answered for a thorough evaluation of a PS version of HSII.

1. INTRODUCTION

A prime candidate organization for large knowledge-rich systems is that of a production system (PS) [10,2,15]. PSs are rule-based architectures that have been used successfully for tasks ranging from models of human behavior [11], to large application systems in chemistry [3] and medicine [13], to general artificial intelligence programming tasks [12]. The uniform rule structure of knowledge in PSs makes them particularly useful for tasks requiring iterative upgrading of knowledge content.

The research question reported on in this paper* is whether a PS architecture (PSA) is adequate to represent the rich variety of speech knowledge contained in Hearsay-II, (HSII), a large artificial intelligence system for understanding speech, developed at Carnegie-Mellon University [1]. We consider explicitly the questions of representational adequacy, space efficiency and time efficiency.**

HSII is a system organization for the speech understanding task designed to coordinate the contributions of diverse, essentially independent sources of knowledge which operate at the different hierarchical levels of speech knowledge. The essential device for communication among knowledge sources (KSs) is a large shared working memory called the *Blackboard*, which all KSs inspect and modify. The Blackboard is a medium for growing linked networks of hypotheses about what elements exist at the various levels and for various time intervals of the utterance (e.g., words, syllables, etc.).

The KSs are structured to contribute their knowledge by creating and modifying Blackboard hypotheses and links in response to relevant changes by other KSs (i.e., in a data-directed fashion).

To carry out the research reported here, an actual production system (called HSP, for "HearSay-Production system") was implemented on C.mmp, the CMU multi-miniprocessor [16], with a portion of the HSII speech knowledge translated into productions. Two knowledge-source (KS) programs from a full HSII configuration were completely translated and run in HSP, and these provide a basis for some detailed comparisons between HSII and HSP. These KSs were (1) POM [14], which recognizes syllables from speech segments, using a complex, multi-phase process involving three intermediate levels of representation; and (2) RPOL [1], which propagates validity ratings for the Blackboard hypotheses along the links that interconnect them. POM and RPOL together required 945 HSP productions. Ten other KSs were translated, but with some simplifications and omissions, and were never completely debugged and run. However, their static structure does provide evidence for representation issues. The total number of HSP productions for all twelve KSs was 2211. If this count is adjusted to discount highly similar productions that simply enumerate cases for a single situation, the result is 293 substantially different productions out of the total of 2211.

2. THE PROMISE OF PRODUCTION SYSTEMS

A production system architecture (PSA), in the sense used here, consists of a set of rules, called productions, stored in a production memory (PM), plus a working memory (WM) which holds symbolic structures operated on by the productions. A production consists of a condition part which tests the current state of WM, and an action part which specifies additions, modifications and deletions to WM to take place in case the condition part is true. The basic operating cycle of a PS is a "recognize-act" cycle: i.e., recognize which production

*This research was sponsored in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-78-C-1551.

**The Ph.D. thesis that this paper is based on [6] considers these same questions in greater detail, plus the questions of exploiting parallelism on multiprocessors and of decomposing a system to run on processors with a small address space.

conditions are true of the current WM, then execute the corresponding production actions. Since each cycle makes some change to the WM state (which changes the set of productions that are true), iteration of the cycle produces (potentially, at least) an interesting stream of behavior.

PSAs at large show a great deal of diversity. The strain to which HSP belongs has a lineage beginning with studies of human problem solving [11] and a PSA called PSG [10], and extending to the more recent architectures of PSNLST [12] and OPS [4]. A variant strain with roots in performance-oriented, knowledge-based expert systems (DENDRAL [3] and MYCIN [13]) embodies a major architectural difference in the use of rules. These systems treat rules as declarative knowledge structures to be interpreted; i.e., the production rules are only a piece of the total system (though the most important piece). On the other hand, PSG-like systems have no declarative long-term knowledge; all knowledge is encoded as (procedural) productions, which combine knowledge with how it is to be used (control information). The tradeoff between the two views is essentially this: PSG-like systems have more flexibility in representing control information (i.e., knowledge interactions), but at the cost of greater difficulties than MYCIN-like systems in augmentation and permitting multiple use of knowledge.

PSAs show substantial promise for large artificial intelligence systems (specifically, speech understanding systems). In [6] this promise is enumerated for the following set of concerns: *adequacy of representation, space efficiency, time efficiency, directionality, augmentation, performance analysis, testing, debugging, and handling of error*. Unfortunately, adequacy, space efficiency, and time efficiency (the three aspects reported on here) are probably the least promising of those listed. The more promising aspects were not within the scope of the research, though some of them are mentioned in Section 4 on "Remaining Questions".

3. THE RESULTS

For sake of clarity, the results of the research are presented in the form of assertions. The assertions are stated in a strong form. A discussion following each assertion explains it and summarizes some of the supporting evidence. In cases where the evidence is weak, the assertion is qualified appropriately. Figure 1 gives a preview.

1. HSP productions are adequate for representation of all HSII speech knowledge.

The basic evidence for adequacy* is that a large number (12) of HSII KSs were translated to HSP productions. There are nearly 20 other KSs which have been used at some time throughout HSII's development, but a moderate acquaintance with them has turned up no reasons to doubt HSP's adequacy. Efficiency is another matter (e.g., see assertion 11).

*By *adequacy* we do not mean theoretical adequacy (which is trivially present), but rather something we might call "representational practicality". We also do not mean to include time efficiency, since that is considered separately.

Representation and Architecture

1. HSP productions are adequate for representation of all HSII speech knowledge.
2. The adequate architecture of HSP is simple.
3. Translating HSII declarative knowledge to HSP creates problems of multiple use.
4. Explicit conditionality on changes is a strong feature of HSP's architecture.

Space Efficiency

5. Procedural HSII knowledge decreases slightly in size when translated to HSP.
6. Declarative HSII knowledge increases in size by up to half an order of magnitude.
7. Total HSII knowledge increases in size by up to half an order of magnitude.
8. HSP requires a much larger global working memory than HSII.

Time Efficiency

9. The implemented time efficiency mechanisms in HSP are critical for its viability.
10. The time cost of HSP relative to HSII is at least two orders of magnitude.
11. Local working memory and control are the prime sources of HSII time efficiency.

Figure 1: Results in Assertion Form

There were several minor difficulties encountered in the KS translation process, for example: iteration over WM element list fields, controlling duplicate actions, tallying events, and redundant arithmetic expression evaluation. Yet these difficulties are not serious enough to constitute a refutation of HSP's adequacy. Most of them could be solved through design iteration of the HSP architecture, without deviating from basic PS philosophy.

HSII KSs do a significant amount of simple table lookup from local arrays containing long-term knowledge. This might at first seem to cause a representation problem for HSP, but actually an array translates cleanly into a mutually exclusive set of productions, one for each entry. Direct indexing to select an array entry in HSII is essentially the same as selecting the correct production from the set in HSP. In other words, a PS is really a large, complex table lookup, while an array access is just an optimization of the lookup process that is possible with a highly uniform set of productions. However, a PSA is overly general for simple table lookup; the resulting space and time costs are discussed below under assertions 6 and 10.

The adequacy of HSP for speech understanding provides

an additional data point for the broader question of adequacy of PSAs for artificial intelligence applications. More extensive evidence for adequacy has already been reported by Rychener [12], in his study of six classical artificial intelligence programs. The speech understanding task of HSP has some characteristics that set it apart from Rychener's tasks: (1) a rich set of relatively independent knowledge sources, (2) each of which has a sound theoretical basis, providing lexicons of basic entities and rules relating entities, and (3) the large direct recognition component, i.e., no need for extensive serial reasoning.

2. The adequate architecture of HSP is simple.

This assertion is of interest only in the context of assertion 1. Simplicity without assurance of adequacy is trivial.

Contrary to standard architectural practice for PSs, HSP has no *conflict resolution*; i.e., on each cycle it fires *every* true production rather than using a conflict resolution mechanism to select a single production to fire. This permits in HSP a natural expression of the multiple, independent KSs that come from HSII and the speech task. Perhaps more importantly, it allows much greater parallelism: while PSG, PSNLT and OPS respond to only a few changes and fire only a single production each cycle, HSP can respond to hundreds of changes and fire a hundred productions or more. However, such a feature could seriously impair a PS's ability to switch focus rapidly in response to a novel situation or new external stimulus [7]. But this does not seem worrisome for HSP since most of the multiple firings are operating in parallel on separate levels or time regions of the representation of the speech utterance.

In the absence of conflict resolution, HSP must occasionally resort to a somewhat inflexible mechanism called an *n-cycle delay*. This involves a chain of productions that waits for other activity to finish by marking time for a certain fixed number of cycles.

HSP permits neither disjunctions nor negated conjunctions within production conditions. (Negations of single condition elements are essential and of course permitted). These restrictions simplify the production interpreter without seriously affecting adequacy of representation. Disjunctions or negated conjunctions can be eliminated from a production by splitting it into several productions, at the cost of some decrease in space and time efficiency.

HSP has no mechanism for *special case inhibition*; i.e., preventing a true production from firing when another production representing a special case of the first one is also true. Such a mechanism would have complicated the production interpreter and probably caused a serious reduction in parallelism. Doing without special case inhibition is an inconvenience. Either the more general production must be augmented to make it complementary to the related special cases, or special productions must be added to detect when both general and special case productions fire and favor the special case result.

3. Translating HSII declarative knowledge to HSP creates problems of multiple use.

A basic feature of the HSP architecture is the absence of

any form of declarative long-term memory. Thus all long-term knowledge must be encoded as productions. Since each production specifies under what conditions its piece of knowledge is to apply, there is a problem with using that knowledge under different circumstances. In some cases the problem can be solved by merely duplicating the knowledge, with a different production for each different use. This was done frequently in the HSP KSs, but will not extend to systems of growing complexity where multiple use can be expected to increase. Subroutines of productions provide another solution to the problem of multiple use. But subroutines require a high degree of similarity of the uses, plus rigid conventions for communication. A third, somewhat novel, solution employed in HSP is to deposit knowledge temporarily into WM whenever there is a reasonable expectation that it may be useful. In WM it is then available in declarative form to whatever production wants to make use of it. It is not known whether these solutions to the problem of multiple use would be adequate for much larger and more complex systems than HSII, but the suspicion is that they would not.

4. Explicit conditionality on changes is a strong feature of HSP's architecture.

The first condition element of every HSP production explicitly tests the nature of a WM change, and must match a change made in the previous cycle. Thus a production cannot fire any time its condition (excluding the first element) is true, but only when a particular type of change (tested by the first condition element) occurs in conjunction with a true condition. (Normally, the change causes the condition to become true). This explicit condition on a change has two important uses in HSP: (1) It provides the basis for a time efficiency mechanism called the *PM index* (see assertion 9). (2) It solves the *excitatory instability* problem; i.e., a production cannot continue to fire cycle after cycle once become true, because it is also conditional on the occurrence of the change that made it true.

5. Procedural HSII knowledge decreases slightly in size when translated to HSP.

Detailed space analysis of the POM KS shows that about 160 Kbits of long-term HSII procedural knowledge translated to only .7 times that much in HSP. There are two possible explanations for this decrease: (1) Since HSP productions are interpreted, a more compact representation is possible; and (2) HSP can represent condition testing and searching of the global working memory more concisely. However, the explanation is probably more complex than this, as suggested by high variation of the HSP/HSII space ratio over eleven subparts into which POM was partitioned for analysis.

6. Declarative HSII knowledge increases in size by up to half an order of magnitude.

HSII declarative structures are mostly either simple arrays, dictionaries of spellings, or linked networks, although all are encoded internally as arrays for efficiency. Arrays translate to HSP as one production per entry; the networks are typically represented by one production for each transition, or one production for all transitions out of each state. In any case, the number of productions required is large.

Seven instances of large declarative knowledge structures in the translated KSs were compared for size. Simple arrays require almost a factor of 4 more space in HSP, but sparseness can be exploited to reduce that factor by the fraction of non-default array entries. Tables of spellings (actually arrays of strings) have HSP/HSII space ratios of about 1. Two examples of network structures have the following ratios: 1.2 (grammar) and 3.2 (state transition network). The overall HSP/HSII ratio for the seven instances compared, excluding one exceptional case (a huge bit matrix), is 1.6.

HSP facilities for *condition and action procedures* allow a sequence of similar condition or action elements to be packaged as a parameterized procedure and then referenced in many different productions with appropriate parameters supplied.* These procedures are crucial for representation of HSII declarative structures because of the large number of similar productions involved. In the HSP POM KS, condition and action procedures give a savings of a factor of 10 in the number of condition and action elements that must be represented explicitly.

7. Total HSII knowledge increases in size by up to half an order of magnitude.

In HSII POM, with an overall HSP/HSII space ratio of 1.1, the declarative/procedural split for long-term knowledge is .3/.7. But this is not a typical split. Many of the large KSs most recently added to HSII are estimated to be split about .9/.1. Assuming a .9/.1 split for a full HSII configuration would give an overall HSP/HSII ratio of 1.5, assuming the declarative HSP/HSII ratio to hold at the value of 1.6 obtained above. If instead we assume a declarative ratio of 3.8 (the worst observed), we get an overall ratio of 3.5. We cannot be more precise than this because the declarative structures of the new KSs have not been analyzed. Their ratios may well be larger than 3.8.

8. HSP requires a much larger global working memory than HSII.

Many HSII KSs use large local working memories in addition to the global Blackboard, and these local memories can be highly specialized for efficiency. HSP has only its global WM, and specialization of it is strongly limited by requirements of generality and uniform accessibility. This difference in specialization costs HSP less than half an order of magnitude in space for simple data items. For more complex data structures such as network nodes the cost can be as much as a full order of magnitude. Furthermore, such complex structures are so abundant that they dominate the overall cost.

9. The implemented time efficiency mechanisms in HSP are critical for its viability.

Three existing time efficiency mechanisms in HSP give a combined speedup of 3 or 4 orders of magnitude over a naive implementation (i.e., one that evaluates every

production as a result of every WM change, and that searches WM for every condition element evaluated). All three mechanisms are suggested by analogs in the HSII architecture.

The first, called the *PM index*, reduces a linear dependence of execution time on PM size to sublinear by associating subsets of relevant productions (i.e., exactly those to evaluate) from classes of WM changes. In the HSP POM + RPOL run this resulted in a speedup of about 200.

The second mechanism, use of explicit pointers between WM elements, reduces the amount of WM searching during condition element evaluation. This mechanism does not reduce the degree of execution time dependence on WM size, but does give a constant factor of roughly 10 to 50 overall speedup.

The third mechanism, an index into WM according to the first two fields of a WM element, serves to reduce WM searching costs. Since it operates in the shadow of the second mechanism which greatly reduces the necessity for WM searching, the WM index makes only a small contribution.

These mechanisms bring HSP to roughly the same level of time efficiency as PSG with filters [8], PSNLST, and OPS.

10. The time cost of HSP relative to HSII is at least two orders of magnitude.

In equivalent uniprocess runs of the POM + RPOL KS configuration, HSP took 255 times as long as HSII (917 sec as opposed to 3.6). This factor of 255 reduces to a range of 6 to 36 when corrected for eight underlying system differences: execution rate of machine, instruction set of machine, address space size, operating system, implementation system, degree of system kernel optimization, speech knowledge content, and complications of parallelism.

There are problems with the generality of this comparison since it is based on a single run of a small configuration (i.e., containing only the two KSs: POM and RPOL), and on only a single syllable of input utterance. Correcting for syllable length and atypicality of the KSs gives a rough overall time efficiency loss of two to three-and-a-half orders of magnitude for a full configuration of KSs in HSP.

Related data by Rychener [12] shows a factor of 6 to 10 loss in time efficiency for PS translations vs. original versions of six classic artificial intelligence systems.

11. Local control and working memory are the prime sources of HSII time efficiency.

Since HSP has such limited local control (its actions are simple sequences and cannot call other productions), it must rely on data-directed invocation operating from the global WM. Much of the data-directed invocation might be avoided if productions could be much larger, compressing multiple PS cycles into single ones. But this is made difficult by an accompanying blowup in the number of productions.

The cost of data-directed control in HSP has several sources: (1) the creation/deletion of change elements, (2) creation/deletion of control signals in WM, (3) PM

*These procedure facilities are purely for space efficiency. A production using them behaves exactly as if all its elements were explicitly written out, except for a small time efficiency loss in assigning parameters to variables.

indexing (finding productions to evaluate based on the changes), and (4) an initial portion of condition evaluation for each production that is necessary to reobtain the context of the preceding production. The total cost of these is estimated to be 30-45% of execution time. In HSII the cost of data-directed control is 9% of execution time, and would be much smaller except that the cost is dominated by the monitoring of Blackboard changes rather than by the number of actual invocations.

The other side of the coin from local control is local working memory use. HSII makes heavy use of local working memory for KS efficiency. Since a PSA contains no analogous facility, HSP is forced to use its global WM for such functions. This puts HSP at a serious disadvantage. The HSP run of POM + RPOL made over 5 times as many global working memory reads as a corresponding HSII run, and more than twice as many creations. Further, as mentioned above, there are other KSs that make much heavier use of local data than POM and RPOL. We expect HSP versions of these others would make hundreds of times as many global working memory reads and creations as the HSII versions.

Two other sources of HSP inefficiency are identified: (1) the absence of a declarative long-term memory facility, and (2) searching of WM. The former consumes 15% of execution time in the HSP run (but some of this overlaps with the WM access costs discussed above). The latter is insignificant (3%) because of the explicit WM element references, but is projected to increase to around 30% of total time with a full input utterance.

Taking all the sources of inefficiency together, we can account for roughly a factor of 7 in execution time. This takes us well on the way toward explaining the normalized HSII-HSP difference of 6 to 36 obtained for the HSP run. But it also suggests there may be other sources. One such possibility is the limited power of the HSP production language, as exemplified by the inability of a single production to deal with a data list of arbitrary length.

4. REMAINING QUESTIONS

From the vantage point of the research done, several questions emerge as the important ones yet to be answered for a thorough evaluation of an HSP-like version of HSII. Six such questions are discussed briefly below. Given the current state of the HSP-HSII comparison (i.e., considering only adequacy and efficiency), with HSP faring rather poorly, it is natural to ask about factors that might swing the balance back toward HSP. Questions (1) through (3) below are of this nature. Questions (4) through (6), on the other hand, ask about possible weaknesses of HSP that need to be probed further.

1. *Do there exist situations that require the inhomogeneity of representation in HSP?*

The HSP architecture is better suited than HSII to representation of inhomogeneous knowledge, e.g., knowledge containing many special cases. HSII's strong point is homogeneous encodings; when it is forced out of these it must fall back on more expensive and ad hoc representations. This could turn into a strong advantage for HSP if the drive for improved performance pushed in the direction of inhomogeneity, as we suspect it may.

2. *Can a significant improvement in ease of augmentation be realized in a PSA?*

Ease of augmentation is one of the most promising aspects of a PSA, so we need to show a significant advantage here to balance negative aspects such as efficiency. HSP's current architecture, lacking conflict resolution, may not properly support augmentation. If necessary, conflict resolution can be added to HSP, at the risk of sharply reduced parallelism.

3. *Is the level of individual productions right for performance analysis?*

Performance analysis at the level of productions can be accommodated easily in HSP. The only issues are whether that level is a useful one (it is perhaps too low), and whether performance analysis at higher levels (always necessary) is made more difficult in a PSA. These issues can be resolved only by experience with larger KS configurations in HSP; the current two-KS configuration is too small to provide any significant insight.

4. *Do there or can there exist HSII KSs which cannot be represented adequately in HSP?*

Since there are about 20 HSII KSs that have not been translated to HSP, plus a virtually open-ended set of others which might conceivably be added to HSII, our evidence for adequacy is incomplete. Particularly suspicious are some of the most recently added HSII KSs that use highly specialized local data structures. The current belief is that HSP is adequate for these, though inefficient. A more thorough HSII-HSP evaluation requires evidence for this belief.

5. *Can HSII KSs that heavily rely on local efficiency ultimately be made tractable in HSP?*

We have seen that the bulk of HSP's efficiency handicap relative to HSII comes from HSP's lack of local working memory and control. It is important to make some inroads on this handicap if HSP is to be useful, and several possibilities present themselves: (1) optimization in the HSP kernel of data-directed invocation and WM access mechanisms, (2) permitting productions to have complex actions with their own conditionality and local working memory, and (3) compiling many PS cycles into one by combining production sequences into many complex productions that act within a single cycle.

6. *Is the distributed control of directionality as dictated by a PSA feasible?*

Claims have been made that PSAs are unsuitable for representation of directionality control [9]. Thus it is incumbent upon us to prove them wrong if HSP is to survive its comparison with HSII. HSII has a specialized focussing mechanism which operates with global knowledge of the state of the computation [5]. In HSP we must show (if possible) that the requisite global state information can be represented in WM, and continually updated by a set of special added productions; and that task productions can be made to schedule themselves with additional condition elements sensitive to this representation of global state.

5. CONCLUSION

We have shown that the HSP architecture is almost

surely adequate for representing the HSII speech knowledge. This includes HSII declarative knowledge which must translate to procedural form in HSP. Adequacy of HSP was not a foregone conclusion because the simplicity of the HSP architecture compared to HSII gave grounds for some doubt about adequacy. Space and time efficiency are another story. The moderate space penalty for representing declarative HSII knowledge in HSP is cause for concern since HSII does contain many large declarative knowledge structures. Even more serious concern arises over space inefficiency of the global HSP working memory, since it must be used in place of large, highly optimized local working memories that are typical in HSII KSs. HSP's lack of local working memory also causes a large loss of time efficiency because of greater creation/read/write costs and heavier use of relatively inefficient data-directed control. HSP's large time efficiency handicap (two to three-and-a-half orders of magnitude) exists in spite of efficiency mechanisms which make HSP comparable in time efficiency to several other PSAs.

The partial evaluation of a PSA for HSII reported here is unfavorable for the PSA, but there are still other aspects (such as augmentation or performance analysis) that hold promise for PSAs. At least we have quantified the efficiency aspects so that a complete picture can be fit together as additional questions are answered.

A side benefit of this research is in providing enhanced understanding of HSII, a system which is a highly visible and important contribution to artificial intelligence. By investigation through HSP of marked alternatives to HSII's philosophy, we shed light on it. For example, we have contributed a better understanding of HSII's use of local memory and control for efficiency.

Another result of this research is the existence of another data point for the applicability of PSAs to artificial intelligence systems. The significant aspect of speech understanding that distinguishes it from the domains studied by Rychener [12] is the heavy use of declarative knowledge structures.

HSP demonstrates several novel features that are of possible interest to designers of PSAs. We have shown that permitting multiple firings per cycle of the PS has appeal (at least for some tasks), and that eliminating conflict resolution does not necessarily cripple; that writing productions to contain explicit conditions on changes is a simple way to avoid repeated firing, and has efficiency benefits as well; that an attribute-value structure for WM and condition elements adds a useful bit of flexibility; and finally, that allowing WM elements to contain explicit references to each other dramatically cuts WM searching during evaluation.

Finally, we have shown that it is possible to obtain meaningful comparisons of related but different complex systems. The field of artificial intelligence could benefit from more comparisons of this sort.

REFERENCES

- [1] CMU Speech Group (1977), "Speech understanding systems: Summary of results of the five-year research effort", (second version), Technical Report, Computer Science Department, Carnegie-Mellon University.
- [2] Davis, R. and J. King (1975), "An overview of production systems", Report STAN-CS-75-524, Computer Science Department, Stanford University.
- [3] Feigenbaum, E. A., B. G. Buchanan and J. Lederberg (1971), "On generality and problem solving: a case study using the DENDRAL program", Machine Intelligence 6, Edinburgh University Press.
- [4] Forgy C. and J. McDermott (1977), "OPS: a domain-independent production system language", Proc. 5th IJCAI, pp.933-939.
- [5] Hayes-Roth, F. and V. Lesser (1977), "Focus of attention in the Hearsay-II speech understanding system", Proc. 5th IJCAI, pp.27-35.
- [6] McCracken, D. (1978), "A production system version of the Hearsay-II speech understanding system", Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University.
- [7] McDermott, J. and C. Forgy (1978), "Production system conflict resolution strategies", in [15].
- [8] McDermott, J., A. Newell and J. Moore (1978), "The efficiency of certain production system implementations", in [15].
- [9] Mostow, D.J. and F. Hayes-Roth (1978), "A production system for speech understanding", in [15].
- [10] Newell, A. (1973), "Production systems: models of control structures", in W. Chase (ed.), Visual Information Processing, Academic Press.
- [11] Newell, A. and H. A. Simon (1972), Human Problem Solving, Prentice-Hall.
- [12] Rychener, M. R. (1976), "Production systems as a programming language for artificial intelligence applications", Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University.
- [13] Shortliffe, E.H. (1974), "MYCIN: A rule-based computer program for advising physicians regarding anti-microbial therapy selection", Ph.D. Thesis, Computer Science Department, Stanford University.
- [14] Smith, A.R. (1976), "Word hypothesizing in the Hearsay-II speech understanding system", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp.549-552.
- [15] Waterman, D. and F. Hayes-Roth (eds.) (1978), Pattern-Directed Inference Systems, Academic Press.
- [16] Wulf, W.A. and C. G. Bell (1972), "C.mmp -- A multi-mini-processor", Proceedings of the Fall Joint Computer Conference, pp.765-777.