1978

Harpy, production systems and human cognition

Allen Newell
Carnegie Mellon University

CMU-CS-78-140

HARPY, PRODUCTION SYSTEMS AND HUMAN COGNITION

A. Newell
September, 1978

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Presented at the Fourteenth Annual Symposium on Cognition, 2 Jun 78; to be published in R. Cole (ed.) *Perception and Production of Fluent Speech*, Erlbaum (forthcoming).

The research in this paper was supported in part by Public Health Service Grant (MH-07722) from the National Institute of Mental Health and in part from the Defense Advanced Research Projects Agency under contract (F33615-78-C-1551) and monitored by the Air Force Office of Scientific Research.

ABSTRACT

Harpy is a speech understanding system that attained (in 1976) a set of highly demanding specifications laid down in 1971 for the recognition of continuous speech (90% accuracy, 1000 words, several speakers, but restricted syntax and semantics). Harpy is an achievement in artificial intelligence, without regard to psychology. Its success, however, makes it worthwhile to ask for its implications for the psychology of speech perception. This paper explores that issue by performing a *sufficiency analysis*, ie, by constructing a psychological model of speech perception that is faithful to Harpy and then inquiring whether it is acceptable given what we know about human processing capabilities. The strategy for accomplishing this is to select a specific model of basic human cognitive architecture, a production system architecture called HPSA77 that is under investigation independently as a model of cognition; and then to map Harpy into this structure in a way that maintains performance plausibility. The paper (1) presents the production system architecture; (2) presents Harpy; (3) performs the mapping; (4) detours into a consideration of intensity encoding in production systems to solve a problem in the mapping; (5) does the sufficiency analysis; (6) examines what the model says about some human speech phenomena; (7) attempts to state what has been achieved. It seems to the author that a viable and interesting theory of human speech perception has been generated by this exercise, though it has several major difficulties (as noted).

HARPY, PRODUCTION SYSTEMS AND HUMAN COGNITION[1]

Allen Newell

The story is by now familiar: in 1971, following upon the report of a study group (Newell et al, 1971), an intensive effort was launched by the Advanced Research Projects Agency of the Department of Defense (ARPA) to advance the art of speech recognition by computers to handle connected speech. Five years of effort by a small community of organizations led in late 1976 to a demonstration of several speech understanding systems with substantial capabilities (Medress et al, 1976).

These systems, eg, Hearsay-II at CMU (Erman & Lesser, 1977) and HWIM (Hear What I Mean) at BBN (Woods et el, 1976), were cast mostly within the main stream of artificial intelligence (AI) systems. They followed directly on the initial image of how to formulate the recognition of speech using multiple sources of knowledge as a problem of heuristic search, though they explored different issues within that conceptual frame. One system, Harpy at CMU (Lowerre, 1976; Reddy, et al, 1977), proved to be the dark horse. Conceived originally by Jim Baker (1975) as a Markov process, the original system was christened Dragon -- to indicate that it was an entirely different kind of beast from the AI systems being considered in the rest of the speech effort. Harpy turned in a superior performance, significantly better than either that of Hearsay-II or HWIM. Harpy was the only one of the systems to meet a set of prominent performance specifications laid down in the original study group report (though Hearsay-II came within shooting distance). Harpy is generally viewed as an engineering oriented solution to the speech understanding task, as opposed to an artificial intelligence oriented solution. (This is not necessarily the view held by its designers and friends around CMU.) In any event, Harpy appears the likely source of a generation of applied limited speech understanding systems for connected speech.

Harpy provides a fine opportunity to play out the scientific tactic of *sufficiency analysis*. Here we have a system, bred out of technology, which accomplishes an important human intellectual function. Harpy, as it exists, has limitations and its extension is an open question technically. But it has so advanced the art of recognition in an area deemed exceedingly difficult for many years, that the mechanisms it embodies rate being taken seriously.

Sufficiency analysis is based on the following proposition:

> Important confirming evidence for a psychological theory is whether a system designed according to the theory is sufficient to perform

1 I am grateful for comments on an earlier draft by John McDermott and Don McCracken. I am also grateful for many discussions on Harpy with Raj Reddy.

> the intellectual functions the theory purports to explain, providing that the mechanisms involved are reasonable according to general knowledge of human capabilities.

Psychology generally ignored questions of sufficiency prior to the development of artificial intelligence, which produced systems that could perform various intellectual tasks. The usual form of sufficiency analysis is to start with an AI system that does perform some task, and analyze whether its mechanisms are reasonable in the light of general psychological knowledge. For example, modern chess playing programs (Slate & Atkin, 1977) examine several thousand positions per second, which is orders of magnitude beyond what humans can do; thus, although they are sufficient for good chess play, they are not acceptable as a theory of human play. Sufficiency is only one type of evidence, but one of substantial power when dealing with complex processes of unknown character.

This paper, then, is an attempt to take Harpy seriously as a model of human speech perception. As a psychological model, Harpy comes unrecommended by its developmental history. Many may prefer to trust the inference from motive: If Harpy was developed without regard to its psychological relevance, indeed with exclusively engineering concerns, then it must have little to say about psychology. My trust is in the opposite inference: that the structure of the task under realistic constraints dictates many features of the mechanisms that cope with it. In any event, the game here is to take Harpy seriously and see where we arrive. No elaborate defense of the methodology of sufficiency analysis is required.

We are not without existing basic models for speech perception, at least in general outline (Studdert-Kennedy, 1976). The two most prominent are the Motor Theory developed by the scientists at the Haskins Laboratory (Liberman et al, 1967) and the scheme of Analysis by Synthesis, initially put forth by Halle and Stevens (1962). Both models build on the indubitable fact that the character of the speech signal is shaped strongly by the system that produces it. Other schemes are not so strongly influenced by motor concerns, such as the Logogen model of Morton (Morton and Broadbent, 1967) and the proposals of Cole and Jakimik at this conference. The theory to be put forth here is not motivated by any of these, being generated directly from Harpy. What other theories it resembles ultimately remains to be seen; it is not specifically designed to be different.

Though interested in Harpy from its inception, not until I read a paper by Dennis Klatt (1977) reviewing the ARPA Speech Understanding Program where he listed some psychological conjectures based on Harpy, did I realize how important it was to try a sufficiency analysis. Dennis is speaking for himself at this meeting, though he has far more credentials than I to attempt the task of this paper. In any event, I agreed to try my hand at it, believing it to be an important exploration, whether or not the resulting psychological theory looks plausible.

I have an abiding interest in a class of system architectures called production systems, both from a psychological viewpoint (Newell & Simon, 1972; Newell, 1973) and an AI viewpoint (Newell, 1977; Rychener & Newell, 1978). It had always been an idea to look at

Harpy from a production system viewpoint, as one approach to sufficiency analysis. Indeed, it also seemed interesting to look at the Hearsay-II organization in terms of production systems (though not with any psychological concerns) and a thesis has just recently been completed along such lines (McCracken, 1978). But not until some recent progress this fall in developing a new production system architecture for human cognition did this connection became strong enough to demand explication of Harpy in production system terms. What started out to be a general sufficiency analysis has become the more narrow enterprise of mapping Harpy into a specific production system architecture. The aim and the caveats are still the same, but a specific vehicle has been chosen.

What is gained by mapping Harpy into a production system architecture rather than analyzing it directly? There is a good deal of evidence that production systems are a good system organization within which to represent human cognition (Anderson, 1976; Hunt & Poltrock, 1974; Newell & Simon, 1972). Beyond this, the new architecture is specifically shaped to be a plausible organization that reflects qualitatively human behavior in a range of short-term tasks that have been much studied in psychological literature (eg, the Sternberg, Brown-Peterson, and memory span paradigms). Thus, to map Harpy into this architecture is to demonstrate that Harpy is a plausible model of speech perception. This (desired) result does not quite follow immediately. The resulting system, call it PS.Harpy, needs to be checked on a number of grounds, eg, can full speech be recognized in realistic times. But the mapping will go a long way towards facilitating such an analysis, since the interpretation of the architecture in human cognitive terms is quite definite.

One criticism should be immediately forestalled. Production system architectures are universal computational machines. There is no doubt at all that Harpy can be programmed as a production system. Thus, what counts is the nature of the mapping, especially whether the resulting processing is plausible given the human limits we already know about.

Given the above remarks, the organization of the paper is straightforward. Section 2 introduces the specific production system architecture that purports to be the architecture of the human mind. Section 3 summarizes the essentials of Harpy and lays out the questions a sufficiency analysis should address. Section 4 gives a preliminary analysis of Harpy as a production system. This will raise a major issue, the representation of intensity, which will be the topic of Section 5. Section 6 then presents a refined analysis taking into account the results of Section 5. To this point, then, we have finally produced a PS.Harpy which satisfies (some) sufficiency issues. Section 7 explores briefly how PS.Harpy deals with a few empirical phenomenon about speech perception. Finally, Section 8 concludes with an assessment of what has been accomplished.

2. THE PRODUCTION SYSTEM ARCHITECTURE

In preparation for mapping Harpy into a psychological theory of speech perception, we lay out a proposed structure of the architecture within which human cognition occurs. This architecture is an instance of a class of system organizations, called production systems, which have received a good deal of attention in artificial intelligence (Waterman & Hayes-Roth, 1978) and cognitive psychology (as already noted). In general, production systems need no defense here, though they are by no means generally accepted. The specific version that I will use was only recently developed. In its own way, it is as novel as the material on Harpy. Unfortunately, we will have to present it here as a given, limiting description to its main outlines and not providing justification for details of its structure. A extended paper on this organization, which is called HPSA77 (for Human Production System Architecture, version 1977), is currently in process (Newell, 1978).

2.1. Basic Production System Architecture

Consider a system with the gross characteristics shown in Figure 1. There is a large memory of productions (PM, for *Production Memory*). Each *production* (P) consists of a set of *conditions* (C_i) and a set of *actions* (A_j). The conditions of each production look into a *Working Memory* (WM) of data *elements* (E_k). The data elements are *symbolic structures* of some sort. Each condition is a template of some kind that can ask whether it matches a given data element. A production is satisfied, at a given moment, if all of its conditions find matching elements in the WM. Such a set of matching elements constitutes a possible *instantiation* of the production.

At a given moment, some set of productions is satisfied. In fact, a production may be satisfied in many ways, corresponding to distinct instantiations. The total set of instantiations for all productions is called the *Conflict Set*. From it an instantiation is selected and its actions executed, ie, the selected production is fired. The actions (the A_j) of a production (properly instantiated by whatever variables are bound by the matching), make a sequence of modifications to WM, adding, deleting and modifying elements.

The behavior of a production system is completely specified by this so called *Recognition-act cycle*:

> Match all the productions of the Production Memory against the Working Memory to determine the conflict set.
>
> Select the successful production by a process of Conflict Resolution.
>
> Execute the actions of the resulting production.

The cycle is repeated indefinitely. All of the conditional behavior of this system is expressed in this cycle; the actions themselves are unconditional operations that affect WM.

The total system shown in Figure 1 consists of the central cognitive structure (PM and WM) plus the structures to interact with the external world. Data from the external world flows into the WM and actions that affect the external world are evoked by elements that are placed in WM. Also required, but not directly represented in Figure 1, are capabilities for modifying Production Memory. In current versions this is realized by actions that create productions out of WM elements.

Conflict resolution is governed by several types of rules. Many types of rules exist (McDermott & Forgy, 1977), but three are worth noting here.

The first type is *refraction*, which inhibits a rule from firing a second time on the same data. The need for some type of refraction arises from the tendency of production systems to go into one-instantiation loops. Given that a production fires at one cycle (ie, is satisfied and wins out in conflict resolution), it clearly is primed to fire again unless something changes. The actions can change things, or the architecture can have refractory rules that inhibit repeated firings -- but something is required.

The second type of conflict resolution rule is *recency*, which prefers instantiations that bind to elements that have more recently entered Working Memory. If WM is thought of as ordered from left to right in Figure 1, with new elements entering from the front, then recency means that productions that bind to elements early in the WM preferentially win out. Recency provides a mechanism for focus of attention and continuity of effort. The result of a production that has just fired becomes the most recent element in WM, and some production involving it has a good chance of being the next one to fire.

The third type of conflict resolution is *special case order*, which prefers a production that is a special case of another. Special case order reflects the heuristic that productions conditional on more data (ie, a superset) are based somehow on more information and should be fired first. Special case order seems well suited to adding new productions to the Production Memory (ie, learning), where it permits new, generally more discriminative productions, to dominate existing ones.

The structure of Figure 1 can be mapped onto the standard picture in Cognitive Psychology of human mental structure as follows. The production memory is *Long Term Memory* (LTM). It contains both data and program -- all the knowledge that the human remembers for more that a few seconds. A set of productions is clearly a program (a set of if-then statements). It holds data by taking the actions as the data expressions and the conditions as the access path to the data. The Production Memory is thus very large, holding hundreds of thousands to millions of productions. Working memory corresponds to *Short Term Memory* (STM). It holds data, but only for a short period of time. Limitations on its capacity and duration are such that the human exhibits the sort of short term fragility that psychology has extensively investigated (eg, see Crowder 1976; Murdock, 1974). The data elements correspond to *chunks*, which have structure and can be decoded into component chunks. The recognition-act cycle corresponds to the fundamental cycle of human cognitive activity, which takes of the order of 100 ms, give or take a factor of a few. Recognition is a

parallel process, since it must select the satisfied productions out of the entire Production Memory each cycle. The execution of a single sequence of action confers a serial aspect to the processing. The *sensory* and *motor* aspects are not detailed; but they act concurrently.

We have sketched only the basic features of a production system architecture. Many aspects have been left unspecified: the exact nature of the actions; the types of templates in the conditions; the details of conflict resolution; the forms of data elements; the timing properties of the recognition-act cycle. There are substantive reasons for not being entirely specific, in addition to expository limits. We do not know yet what exact variant of the architecture best describes the human cognitive structure, even assuming a production system to be the correct scheme (a proposition that many may not wish to subscribe to in any event). Thus, a particular variant (HPSA77) will be described below as the starting point for our investigation into Harpy. And even here, many aspects will remain open to be possibly further specified in the attempt to discover how to structure a production system architecture in a Harpy-like way to recognize speech.

There are positive reasons for considering together the class of architectures based in Figure 1. Many variations of such architectures have been tried (Baylor & Gascon, 1973; Forgy & McDermott, 1977; Moran, 1971; Newell, 1973; Rychener, 1976; Young, 1973). Experience shows them to be equivalent in many ways. The common features of Figure 1 does indeed impose a common gross behavior, and the further specifications affect mainly details.

To make these notions concrete, Figure 2 shows a collection of productions which can illustrate a variety of features. They are written in a specific architecture, called OPS2, which we are currently using for our artificial intelligence research (Forgy & McDermott, 1977). The task is a simple Sternberg item classification (Sternberg, 1975), where only the essential features of the task are present, without any controlling context of evocation and response, and without any potentially disruptive additional data elements and productions. The PS has acquired a set of digits, each coded as (Digit ...); they are sitting in WM (shown at the bottom of the figure). The system is to say yes if the probe digit, coded as (Probe ...), is a member of the set, no otherwise. The elements may have arrived in WM from the external world or from the execution of other productions (eg, if the set were being held in LTM).

Consider productions A1 and A2, which form a production system that can perform this simple task. The elements to the left of the arrow, -->, are the conditions. A1 has two conditions, A2 has three. The elements to the right of the arrow are the actions; each production has a single action, A1 to say yes, A2 to say no. The terms with the equal sign (=X, =Y, etc.) are variables, which may be instantiated so as to make the condition, which is a template, match a data element. With the given WM, A1 will be satisfied if =X takes the value 6. The conditions in a production can match elements anywhere in WM, eg, A1 matches to the 1st and 3rd. A2 cannot be satisfied with WM as shown; thus there is only one satisfied production and therefore it fires. If WM contained (Probe 5) instead of (Probe 6), the A1 would not be satisfied and A2 would, leading to saying no. A2 makes use of a negative condition, -(Digit =X), which is satisfied only if (Digit =X) cannot be satisfied. Thus, the

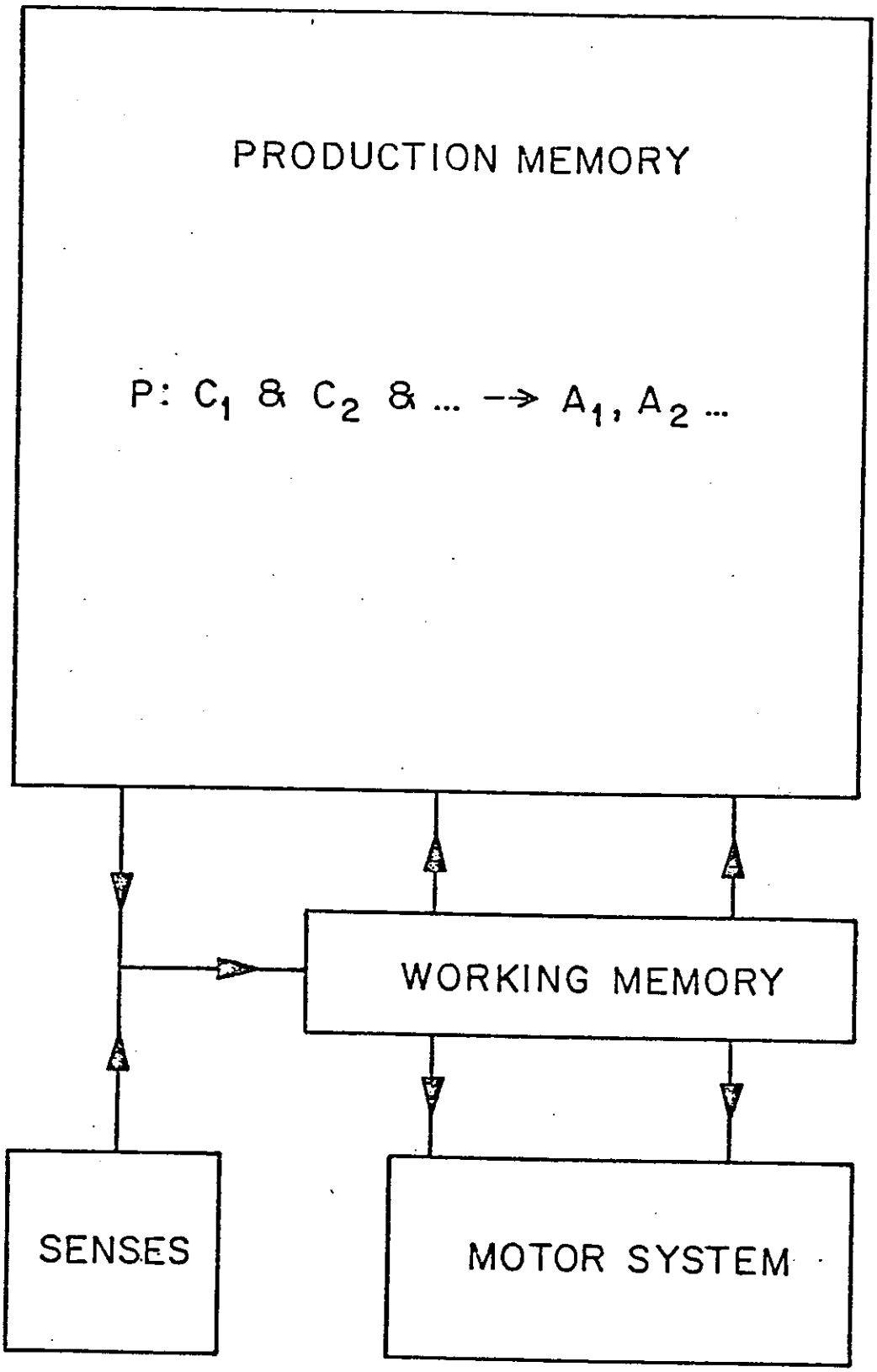


Figure 1. Production System Architecture

A1: (Probe =X) (Digit =X) --> (Say "Yes")

A2: (Probe =X) -(Digit =X) (Digit =Y) --> (Say "No")

B1: (Probe =X) (Digit =Y) --> (Test =X =Y)

B2: (Test =X =X) --> (Say "Yes")

B3: (Test =X =Y) -(Test =Z =Z) --> (Say "No")

C1: (Probe 6) (Digit 6) --> (Say "Yes")

C2: (Probe 6) -(Digit 6) --> (Say "No")

D1: (Probe 6) --> (Say "Yes")

D2: (Probe K) --> (Say "No")

E1: (Probe =X)&=Z (Digit =Y) --> (<Delete> =Z) (Test =X =Y)

F1: (Digit =X)&=Y --> (Say =X) (<Delete> =Y)

WM: ((Probe 6) (Digit 1) (Digit 6) (Digit 4))

Figure 2. Example production systems

recognition match consists of a number of mechanisms (variables and negated elements, here), which collectively define its power.

The need for avoiding loops is apparent with A1. Nothing keeps it from being satisfied on each cycle so that the behavior of the PS would be "Yes Yes Yes ..." indefinitely. OPS2 uses a rule of *absolute refraction*, in which an instantiation can never be executed a second time on WM elements that have not been changed. This effectively stops such behavior. Production E1 (towards the bottom) shows another way of avoiding the immediate loop, namely, by removing one of the elements necessary for the production (ie, E1) to fire. E1 also illustrates the use of an an action (deletion) other than adding an element to WM. It also shows the labeling of an entire condition element with a variable, by &=Z, which is needed in conjunction with the <delete> function. A1-A2 does not directly illustrate the other conflict resolution rules. But if the negation were removed from A2 (and ignoring whether it would make sense), then both A1 and A2 could be satisfied by the WM shown. A2 would win by virtue of special case order, since it would bind to all the same data elements as A1, but also to another one, thus being the more specialized production. Recency would occur if one of the productions bound to memory elements that had entered WM more recently than those for another productions. OPS2 uses a strictly lexicographic ordering on memory elements to determine recency if the productions bind to several elements in WM (as both A1 and A2 would). B1-B3 in Figure 2 provides an alternative PS for performing the same task. Instead of producing the answer directly, B1 creates temporary data elements, (Test ...), which B2 then tests. As matters stand in OPS2, A1-A2 is a much more efficient way of doing the task than B1-B3.

2.2. *HPSA77: Further Specifying a Cognitive Architecture*

In the attempt to discover the production system architecture that best matches the evidence on human cognition, four additional assumptions have been made. Each is a related group of details of the architecture, built to respond to some major type of human behavioral data. We present these with a minimum of discussion, though they all rate extended treatment. They specify the architecture only somewhat further; many additional details still remain open. These four design decisions are intended to be the initial part of a more extended sequence of design decision that will ultimately fully determine the architecture.

On the Serial Limitation and the Basic Cycle Time. The basic structure of a production system architecture is to be a parallel-recognition - serial-action system. From this flows the basic serial character of high level cognition. This first additional assumption delimits the locus of seriality further:

> D1.1. There is a single mechanism available for instantiating, binding and using variables so that only one instantiation involving variables can be executed at a time.

Productions (such as B1 in Figure 2) that contain variables in their conditions that are used

to instantiate expressions on their right hand sides, are very different from productions, such as C1, that contain no variables. B1 must line up to use a limited mechanism, C1 need not. We call this mechanism the *Use-Variables* (UV) mechanism. We leave entirely open the locus of the limitation: whether in the instantiation of the variables in the condition to the WM element; in the binding of the variables to the value for future use; in the copying of the action element; or in each act of replacement of a variable by its value in an action.

Along with assumption D1.1 goes another:

> D1.2. All the instantiations of a selected production are executed at once, before the next recognition occurs.

What is selected by the conflict resolution is one of the satisfied productions and it locks out all other productions until all of its instantiations pass through the Use-Variables mechanism. This makes the actual computational cycle consist of a recognition, a selection of a production, and the execution of several sequences of actions, corresponding to each instantiation of the production. For example, B1 is satisfied in three ways on WM, one for each (Digit ...) element. Thus if B1 fired, it would put three new elements in WM. Note that D1.2 is not a complete specification; it does not specify in what order the instantiations are executed, if that is important.

Since there may conceivably be a large number of instantiations of a single production, there may have to be a limit to the extent of the lockout:

> D1.3. An arbitrary limit may exist to the number of instantiations or actions that may be executed before the next recognition phase begins.

To illustrate, if WM had a thousand (Digit ...) elements, there might be only seven of them that could generate (Test ...) elements before going back to the recognition phase.

A final assumption that limits the power of the recognition match belongs in this group. Each condition can seek an element anywhere in WM that satisfies it. In most production systems variables are freely permitted throughout the conditions. Thus the recognition phase is capable of detecting patterns of elements satisfying various equality conditions among the components of working memory elements. A1 in Figure 2 provides an example, since =X occurs in two separate condition elements and hence only if two separate WM elements are equal on this component will A1 fire. This appears too powerful an ability in the human cognitive system, and we restrict it as follows:

> D1.4. The same variable cannot appear in more than a single condition element.

A variable can of course be used in more than one action element. D1.4 limits only the condition elements, which express the power of the match. However, a variable can occur

more than once *within* a single element. B2 in Figure 2 provides an example, and thus contrasts with A1, which now becomes inadmissible. Multiple occurrences of a variable must occur someplace, if the system is to be capable of performing equality tests.

The assumptions D1.1 - D1.4 were all dictated by the attempt to come to grips with what can broadly be called Sternberg phenomena (see Sternberg, 1975) -- linear dependencies of reaction time on set size with coefficients well below 100 ms per item (the classical Sternberg item classification tasks yields about 40 ms per item). Interpretation of these fast rates as search times through the sets raises the possibility of a computational cycle time of the order of 40 ms. This appears to imply too much computational power in a few seconds. The assumptions grouped under D1 draw the teeth of this implication by offering a different explanation for the Sternberg coefficient, namely that it corresponds to the operation of the UV mechanism. With the duration of UV set at about 40ms per instantiation, HPSA77 provides an explanation for many of the important phenomena in the Sternberg paradigm. For instance, in Figure 2, A1-A2, which would permit size-independent behavior, is not a possible method, whereas B1-B3, which shows linear behavior, is. On the other hand, C1-C2 will also accomplish the task directly and is variable free; the problem is that C1 and C2 cannot be constructed on the fly for each new trial.

Creation of New Elements. The normal assumption in a production system architecture is to permit actions to create new elements in WM and for these to persist for a substantial period of computation. It is easy to show that this ability would provide the human with cognitive powers sufficient to obviate most of his well known short-term memory difficulties. Briefly stated, the only restraint on using mnemonic devices like "one is a bun, two is a shoe, ..." to solve all the short-term memory problems is that the human cannot create the necessary temporary structures sufficiently rapidly and reliably. On the other hand, if no new elements can be created at all in the short term, little computation of any sort can be done. For instance B1 in Figure 2 would be inadmissible.

The solution seems to be to permit actions to create data elements in Working Memory, but to limit their life severely. One alternative is to treat all data elements uniformly. Another, which is taken by HPSA77, is to distinguish *new* data elements from *old* data elements that have already been learned by the system, and to permit different lifetimes for each type. Thus data elements themselves can be said to become *established* in LTM. Since in a production system architecture the only LTM is Production Memory, the set of established (old) data elements corresponds to those that occur as actions of productions. New data elements are then those that do not so occur. They can arise either from instantiations of actions that contain variables or (possibly) from the external world. Note that an action that contains variables is not a data element, since it cannot exist as an element in WM; it is a generator of data elements, some of which may already exist as actions of other production and some of which may be new.

A convenient way to view such a system is in terms of activated elements:

D2.1. Working Memory is the set of activated elements; ie, an

element of LTM *enters* WM when it becomes active and *leaves* WM when it ceases to be active.

This makes WM a set. Multiple occurrences of an element cannot exist in WM, since it only constitutes an activation on the set of elements in LTM. For instance, there could not be a second copy of (Digit 2) in the WM of Figure 2. If repeated digits were given, more complex data elements would be required, such as (Digit 4 first) and (Digit 4 second).

D2.2. Elements that are already established in LTM have relatively long residence times in WM, of the order of seconds.

D2.3. Elements that are not already established in LTM have relatively short residence times in WM, of the order of a quarter of a second.

In terms of activation one can think of established elements continuing to exist in LTM (ensconced) in productions, even though they cease to be active, whereas new unestablished elements cease to exist when they cease to be active. For example, the (Test ...) elements of B1 in Figure 2 would all quickly disappear; they would remain just long enough to do their computational job.

D2.4. The time to create new elements that exist permanently in LTM is of the order of several seconds.

The relative times are still left quite open, especially the longer ones. A new element can arise in the system whenever an action element is created with a variable instantiated to a new value. Thus the UV mechanism does create new elements; they simply do not last long. The question remains open whether new elements can be created from the outside. HPSA77 also remains entirely open at this point on the mechanism for permanent creation, and whether there are separate mechanisms for creation of new elements and for creation of new productions. They are clearly tied together in some respects, but establishment of data elements may involve only very specialized production-creating mechanisms.

With the assumptions of D2 the architecture appears to behave properly with respect to certain short term memory behavior. In the Brown-Peterson (Peterson & Peterson, 1959) paradigm of short-term rapid forgetting under rehearsal-free conditions, HPSA77 has difficulty in remembering the target words, as do humans. Furthermore, its problem is essentially one of interference. It can recall the familiar components it was presented, but cannot differentiate which were actually in the specific stimulus. Thus it shows perfect performance on the initial trial, release from proactive inhibition, and similar phenomena, in qualitative agreement with human behavior (Crowder, 1976).

Forward Order of Experience. It is a commonplace that people recall their experience in the same time order in which they experienced it, ie, in the forward time direction. This creates a difficulty for production system architectures generally speaking, and probably for

many other architectures (but not for all, eg, Estes, 1972). Production system architectures tend to be stack machines, in that WM is ordered with new elements entering at the front. In our exposition of the basic architecture, the recency rule for conflict resolution conveys this characteristic. But this makes the "natural" character of forward experiencing difficult to understand and to implement. For instance, F1 in Figure 2 is the obvious way of repeating back the digits in WM -- but it does so in opposite order. The following assumptions deal with this problem:

> D3.1. Productions themselves remain in a state of activation after the production is executed.
>
> D3.2. Activation of productions lasts a relatively long time, of the order of many seconds (or even much longer).
>
> D3.3. An activated production will become satisfied again if an element enters WM that satisfies a single one of its conditions.

These assumptions imply that if a single element from the past is recalled then it can possibly trigger productions that were executed at the original time, thus iteratively producing a stream of elements running forward in time that reconstruct the original memory as experienced. This will be a genuine reconstruction, since what is reproduced is what could be recognized by productions existing at the time of the experience. Nothing is said about how long the state of activation lasts, except that it is not short-lived. This lack of specification is dictated (as in the other assumptions) by the desire not to specify more of the architecture than is necessary to solve a specific difficulty (here forward experiencing), leaving as much freedom as possible for future issues.

This assumption of activating productions as well as data elements may seem quite radical. Interestingly, it is very close in spirit to the absolute refractory conflict resolution rule use in OPS2. Namely, as long as all the elements that satisfied a production remain in WM, a production will become a candidate for re-execution if a single element to which it binds becomes reasserted. Stated somewhat tersely: Over the period in which WM lasts, D3 is equivalent to absolute refraction.

Some short-term memory reporting phenomena seem to be explained in a qualitatively satisfactory manner with D3. In simple digit span experiments, reporting digits back in forward order is much easier than in backward order. HPSA77 does memory span experiments by recognizing the material on input, thus activating a bunch of productions (ie, those that do the recognition). Recall occurs by first recalling an anchor point (eg, the initial item in the sequence) and then trying to recapture the sequence running forward, by refiring the original recognizing productions. This not only produces appropriate forward recall behavior, but makes backward recall more difficult.

Sensory Processing. Figure 1 shows the main cognitive engine. It is necessary to fix how information from the senses flows into it and how motor actions issue from it. The

sensory aspects are especially crucial for the topic of this paper (though from a motor theory of speech perception, motor aspects may be almost as important). In the usual view of human memory structure a set of sensory buffers (the Sperling memory, the auditory buffer, etc) exist, which lie outside the central system of Figure 1 and feed WM already symbolized data. Thus there is another memory or memory-access system that makes contact between the sensory features and some sort of lexicon converting the features to (verbal-linguistic) symbols, which are then subject to further cognitive processing (the logogen memory is an example; Morton & Broadbent, 1967). HPSA77 makes a different assumption:

D4.1. All sensory contact with cognitive knowledge occurs via WM.

This implies that much of the conversion of sensory information to symbols occurs in WM. D4.1 does not specify where the boundary is, ie, at exactly what place and in what form knowledge is encoded outside the system of Figure 1 and enters WM. What arrives might reasonably be called sensory *features*. It claims only that the knowledge coded in the form of productions in PM can't be used in the preprocessing stages that produce such features. Thus, these features, though they might be symbols or data elements in the WM, cannot yet represent words or other semantic constructs, which are encoded in productions. This seems to imply, and we will take it thus, that the sensory buffers are part of WM.

D4.2. Productions that do not require the Use-Variable (UV) mechanism are free to execute asynchronously and concurrently, subject perhaps to lockout restrictions if common structures are accessed.

This assumption is the other side of the coin of D1.1 and is not really independent of it. Having located the serial constraint in a specific mechanism (UV), if that mechanism is not evoked there should be no serial constraint. The concurrent productions are easily characterized: they do not contain variables. The qualification in D4.2 refers to the possible existence of an entirely separate source of limitation on concurrency, which arises when two active elements wish to touch the same structure. The caveat is included only to acknowledge the potential existence of such constraints, not to make any substantive pronouncement on them.

These assumptions on the sensory system do seem to be quite radical compared with the other assumptions that have been made, at least in certain superficial appearances. For instance, if one thinks of the amount of low-level information involved in perception, then WM must contain hundreds of thousands of active elements, though they may exist there for only a short time. Likewise, many thousands of variable-free productions may be firing simultaneously in the processing of such features, though any time a production is selected that involves variables it must line up in front of the UV mechanism and wait its turn to have its variables instantiated, bound and used.

There are many sensory phenomena to which the D4 group of assumptions is

responsive at least superficially. They involve issues of how the cognitive system penetrates into perception and how various tasks involving perception of sets of elements show size independence, in marked contrast to the linear size dependence shown by tasks such as the Sternberg paradigm. A simple way to indicate the scope of these these assumptions is by reference to a recent theory put forth by Shiffrin and Schneider (1977; Schneider & Shiffrin, 1977). HPSA77 can be taken to be a more explicit version of the Shiffrin and Schneider theory, in which the variable-free productions map into their *automatic processing* and the variable-containing productions map into their *controlled processing*.[2] It is more explicit because it provides a total control structure with common underlying mechanisms, while assigning specific mechanisms to the two types of processing (automatic and controlled), and it offers an explanation for why controlled processing is serial. It is useful here to note the correspondence, because generally speaking HPSA77 will cover the same ground as the Shiffrin and Schneider theory and offer similar explanations for various phenomena.

Summary on HPSA77. The preceding architecture is the starting point for understanding how Harpy can be seen as a model of human speech perception. By its construction HPSA77 is a plausible model for the structure of human cognition. The evidence for this is not detailed here, but is taken as a working assumption. As noted already, HPSA77 is deliberately an incomplete specification, preserving as many degrees of freedom as possible while still explaining various phenomena. For instance, no mechanisms for production creation have yet been posited. Some parts of the architecture may need substantial elaboration when we come to the speech task, but this introduction is sufficient to let us move on to the presentation of Harpy itself and what is implied by a sufficiency analysis of it.

---

2 For instance, productions D1-D2 in Figure 2 illustrate roughly what is possible in a typical situation where automatic processing is possible. Because the probe set and the target set remain disjoint over many trials, variable-free productions such as D1 and D2, which permit direct response, can be acquired.

3. SUFFICIENCY ANALYSIS OF HARPY

Harpy is no more a single specific system than is the basic production system of Figure 1. Starting with Dragon, there has been a continuous tree of evolution with some dozens of variations, some of which have proved abortive and some of which have survived. We need to extract from this variety the "essential Harpy" for our purposes. At this conference Raj Reddy has given a more complete picture (Reddy, this proceedings; see also Lowerre, 1975), so we can be brief.

3.1. Summary of Harpy

Figure 3 gives one picture of the Harpy algorithm. Ultimately, at performance time, there exists a great set of states. Each state has associated with it a phone template for the character of the sound to be heard if in that state, and a set of transitions to states that can immediately follow it in time. Thus a state encodes the understanding that a certain sound has been heard, along with the expectations for what acoustic input should arrive next.

Taking the transitions to have probabilities associated with them and the template comparison process to yield a probability that the phone was heard at that point, the Harpy algorithm can be viewed as finding the maximum likelihood path through the network formed by iterating the state system indefinitely though time at some basic time grain. The actual path equation is:

$$(3.1)\quad P_{i,t+1} = C(A_i,D_{t+1}) * Max_j(P_{j,t} * T_{j,i})$$

$P_{i,t}$ = Probability of state S_i at time t
$C(A_i,D_t)$ = Comparison of A_i, the template for state S_i, with the acoustic data signal, D_t, at time t
$T_{j,i}$ = Probability of state S_i arising immediately after state S_j

Taken in full (as performed by Dragon) the algorithm is a simple dynamic program that sweeps across the entire rectangle of states and time, keeping only the maximum as it goes and cumulating the probability score. At the end (tf), the maximum $P_{i,tf}$ constitutes the final point on the solution, and a backtrace through the rectangle using saved back pointers yields the optimal path, hence the maximum likelihood utterance.

Harpy consists of a number of modifications of (3.1). First, the empirical evidence shows that the transition probabilities can be dispensed with, ie, replaced by 0s and 1s. This simplifies the path equation to:

$$(3.2)\quad P_{i,t+1} = C(A_i,D_{t+1}) * Max_j \text{ in } T_i(P_{j,t})$$

T_i = Transition set of $T_{j,i} = 1$

Now a substitution of logarithms produces:

(3.3) $L_{i,t+1} = C(A_i,D_{t+1}) + Max_{j \text{ in } T_i}(L_{j,t})$

$L_{i,t} = Log(P_{i,t})$

The next major amendment is to introduce *beam search*. Thus, as shown in Figure 4, only a variable fraction of the states (the beam) are kept at each point in the search. This is about 1% for current Harpy and thus represents a significant computational reduction. Restriction to a beam does imply the possibility of missing the solution path, since if one of its intermediate states is poor enough to drop out of the beam the path is gone forever.

With the notion of the beam comes the decision process to determine the cutoff:

(3.4) e = Beam cutoff at t: Reject all $L_{i,t}$ more than e from the maximum.

The exact form of the cutoff is important. Neither a fixed number nor a fixed fraction works well. The current version keeps all states whose likelihoods are within a fixed range of likelihood from the best. This admits a variable number of states to the beam, but the criteria remains constant.

If Harpy has a beam of F states at time t, it undergoes expansion at t+1 to F*B states, where B is the number of transitions (ie, the average size of the connection set, T_i). This expanded beam must then be clipped to bring it back into line. If the best likelihood (L*) were known in advance, then the threshold of (L* - L < e) could be applied as each transition was generated. However, L* is not known with surety until all transitions have been generated; hence a slightly more complex computation is required.

The next essential feature of Harpy is the variable size of the basic time step. This has two parts. First, the grain of the computation was originally fixed (at 10ms). Thus the Harpy algorithm worked directly with continuous unsegmented speech. Currently, Harpy has an independent preprocessing step that segments the signal into variable duration units on the basis of similarity of the parametric representation. Segments run from 30 ms to over 100 ms, and average about 50 ms. The use of a variable segment implies that the beam cutoff threshold cannot remain constant over all segments. Harpy currently makes it proportional to the duration of the segment (ie, e = e' * duration), but this is not wholly satisfactory. There is also a separate determination for each step whether the path through a state will remain in the state or will exit according to the possibilities in the T_i, the transition set. This is done by a mechanism that decides whether the next segment is sufficiently similar to the current one. It can be viewed as an implicit loop-back of S_i to S_i at each point, except that separate criteria are used for proceeding other than that implicit in $C(A_i,D_t)$.

The nature of the comparison between acoustic template and data is that a set of 14 components (LPC coefficients) characterize the speech around the point in question and a metric (called the Itakura metric) is computed between these 14 parameters and a corresponding set of 14 parameters that make up the template. This metric is a general

quadratic form in the parameter vectors, but once the acoustic template is fixed the expression just becomes a weighted sum:

(3.5) $C(A,D) = \mathrm{Sum}_{k = 1:14} W_k * D_k$

There is a fixed set of templates (currently 98), which corresponds to an acoustic definition of phones. These are modified for each speaker on a running basis from words he speaks. Very little data is required for this, so that a running calibration takes only a second or two of speech per phone.

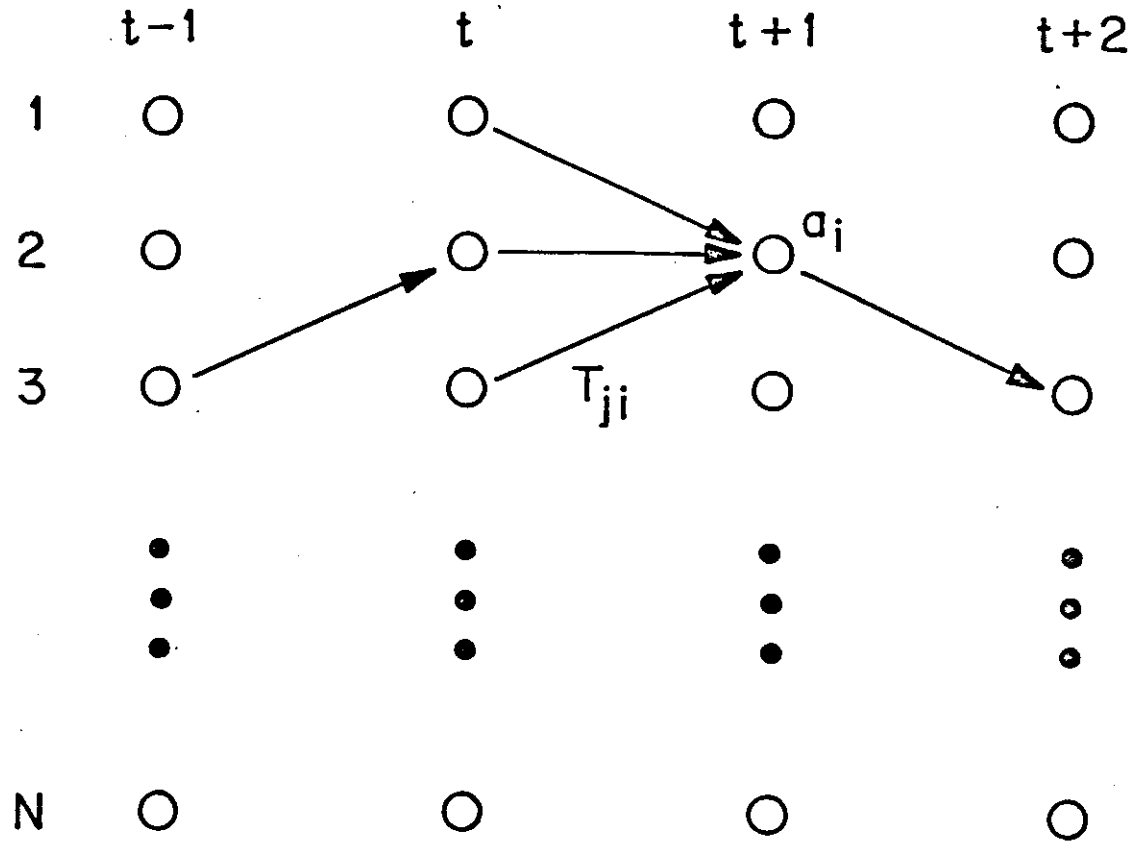
The above summary covers the Harpy performance system. The network (ie, the state system with the transition matrix) does not occur as an act of god, but is derived from language knowledge in a specific way. Given some generative representation of a grammar (Harpy grammars have all been context-free, so called BNF, grammars), a finite grammar network can be generated for any fixed period of time, though it grows combinatorially. Figure 5 shows a fragment of a grammar net, containing several potential utterances, such as "Tell me about China" and "Tell us about all the stories". This net embodies grammatical knowledge. In the Harpy framework, which uses artificial task-oriented grammars, such a net also embodies the semantic knowledge, which has been woven into the grammar itself.

Similarly, each lexical item (word) can be characterized by a network of states, representing which phones can follow which other phones in the various pronunciations of the word. Figure 6 gives an example for the word "please" (the curly brackets give minimum and maximum default durations). These nets embody both lexical and phonological knowledge. Due to the finite character of words these nets have a fixed size, unlike the grammar nets which grow exponentially in the duration of the utterance.

The single-level performance net is created by replacing each node of the grammar tree, which represents a lexical item, with the corresponding net of phones. When this occurs a number of rules must be applied to account for coarticulation effects that arise because of the abutment of the end of one word with the beginning of another. Such rules are not needed within the word network; ie, the net is the final result of having already applied such rules. These rules modify the total net, both adding links and deleting links at word junctures, thus producing a final network that is no longer a simple expansion. In Harpy, this network is produced as a distinct act of compilation. Some additional state minimization operations are performed, which do not change the qualitative performance but increase the efficiency somewhat.

There are a number of minor features in Harpy as it exists, especially to increase its efficiency. There are also a number of major improvements in the works to improve its accuracy. These latter are ultimately of importance, but we can examine the present Harpy as a plausible speech perception mechanism for humans to discover where the critical issues lie.

$$\text{Max}\left[P_{j,t} \cdot T_{ji}\right] \cdot a_{i,t+1} = P_{i,t+1}$$



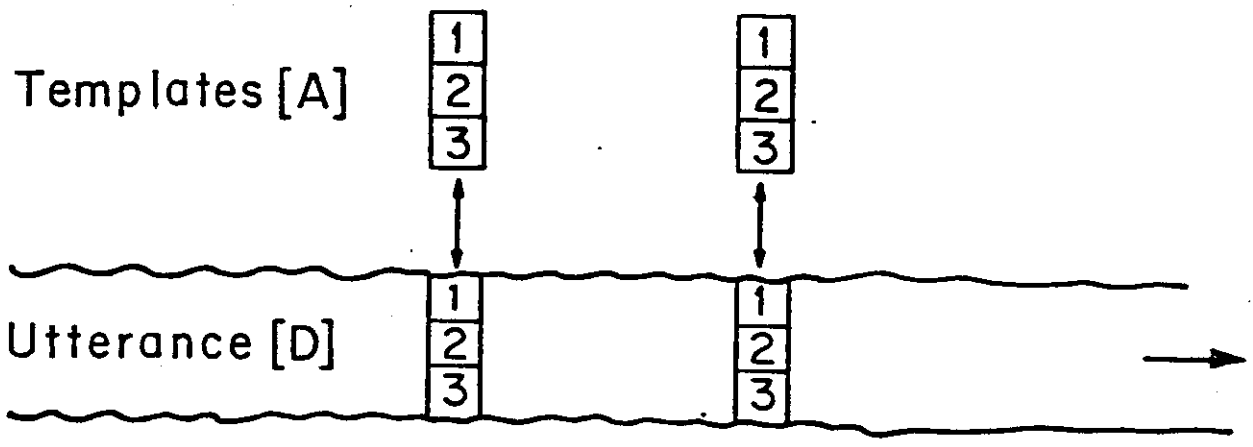


Figure 3. The Harpy Algorithm

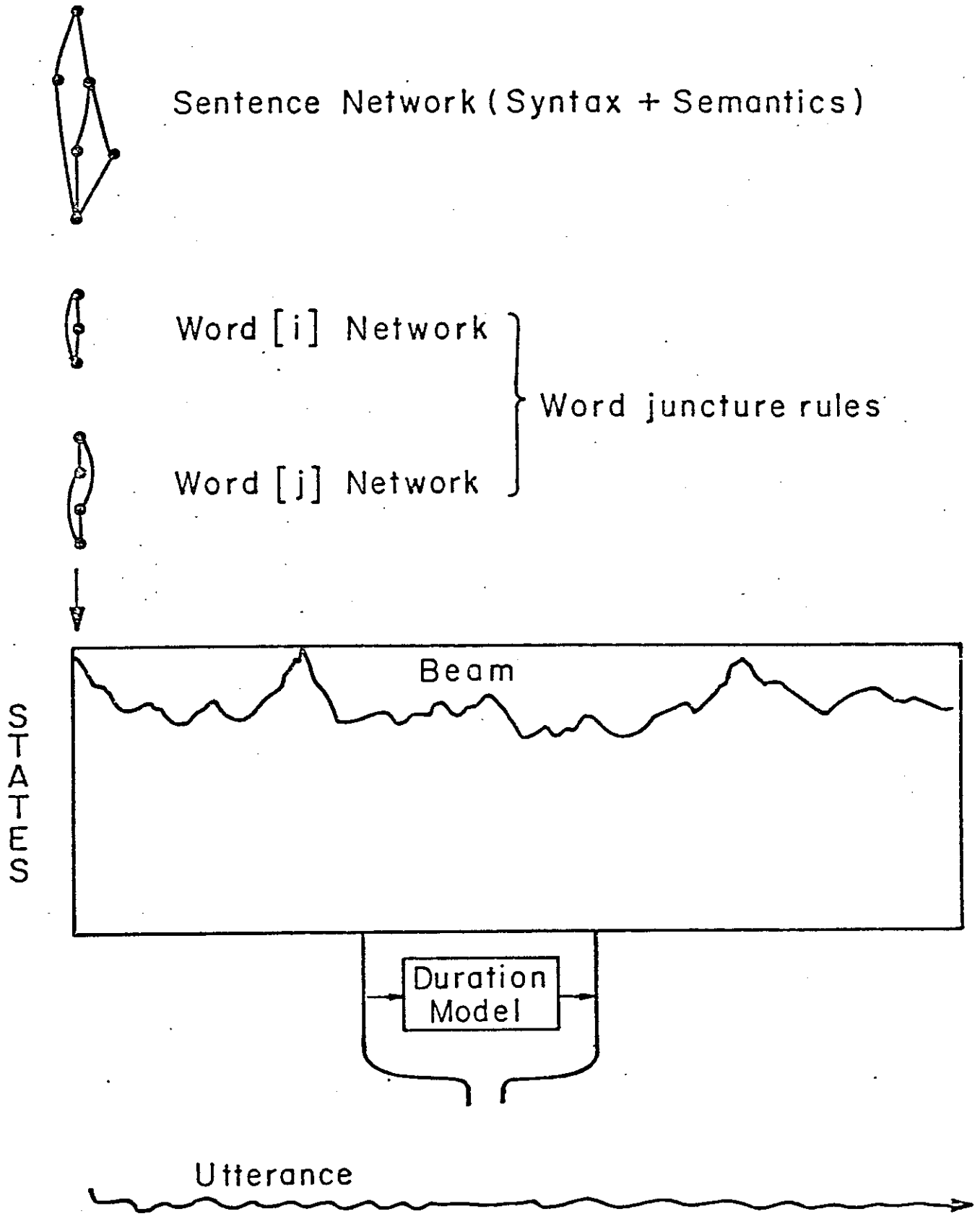


Figure 4. The beam search

3.2 Issues of Sufficiency Analysis

Given Harpy as it stands, we can ask how plausible it is as a model of human speech perception, looking only at its gross characteristics. What seems to fit the requirements of perceiving speech in terms of what we know about humans? What seems to be its most outlandish aspects? Our purpose in this section is to raise these questions rather than answer them. They will sensitize us to the important issues as we try to map Harpy into a production system model of speech perception in the next section. For ease of reference we list the questions in Figure 7.

(S1) Can Harpy recognize speech? A sufficiency analysis starts with a system that can perform the intellectual function in question. Thus Harpy can indeed recognize connected speech, and at a non-trivial level of accuracy and sentence complexity. That is what it means to meet the ARPA specifications of 90% semantic accuracy, 1000 words vocabulary, several speakers, little training per speaker, etc. This is the strong point. However, we must recognize that Harpy is not the only scheme that can recognize speech. Both Hearsay-II and HWIM, as well as a number of other systems (Jelenek, 1975; Nakagawa, 1976; Ritea, 1975), can also recognize connected speech. Differences in capability could be assigned to factors that are not relevant to the psychological plausibility of the systems, such as development effort or whatever. Another system concept could be adopted as a starting point for a sufficiency analysis, say the multiple knowledge source philosophy of Hearsay-II, which has some very attractive features as an AI system. Such a course might well be worth pursuing and is in no way precluded by the present analysis. Indeed, Klatt's contribution to this symposium (Klatt, this proceedings,) draws on many different mechanisms from the entire ARPA Speech Understanding Systems research effort; his strategy may well be the preferred one.

Two generalized features of Harpy recommend it as basis for a perceptual system. The first is the question of search. The real time aspect of speech argues against there being very much open-ended search. However much one might want otherwise, it seems unlikely that a technical solution to the speech problem exists that does not involve search. The uncertainties are too strong. The fundamental character of Harpy lies in its transformation of the search from an exponential process to a linear one. This is accomplished, not by magic, but by fiat, ie, by forcing all the incoming perceptual information into a finite (though large) set of states. An absorption of uncertainty must occur in this forcing, and errors are generated thereby. Whether more or less than in other schemes is not determined just by the choice of search scheme, but involves many other aspects of the total scheme. The gain with Harpy is to bring the search under control, so that search can be safely indulged in as a real-time process. This seems to me a great virtue and recommends Harpy-like schemes over others such as Hearsay-II and HWIM, where the search control comes about in different ways.

The other generalized feature that appeals to me about the Harpy scheme is the uniformity of its structure and the consequent simplicity of its algorithm. Successful technologies for doing complex tasks seem to arise by finding a unit or element that is

capable of restrained diversity, but which can be multiplied indefinitely to attain the complexity desired. The technology of DNA/RNA is an example; so is computer technology, in each of its successive stages. This may just be another way of stating the presumption of hierarchy -- that all complex systems require a hierarchical structure (Simon, 1962). But it seems to add to this an assumption of uniformity and combinational involution within a level, which the usual formulation does not dwell upon. In any event, the uniform structure of Harpy makes it seem like a structure that could indeed arise as a natural technology for perceiving speech.

Having said a few pleasant things about Harpy, let us now move on to the problems and issues.

(S2) Can Harpy be extended to full speech? All speech systems, Harpy included, have only worked with limited language, limited vocabulary, limited grammar, etc. Though a presumption of sufficiency must be admitted, it may not survive as one proceeds to open speech. However crude, it seems worthwhile to make some estimates of how big Harpy would be for full speech.

The first critical parameters are the total number of states, S, the branching factor of the net, B, and the depth that must be searched, D. The accuracy of recognition is also critical; we will implicitly be holding this constant (though the assumption that Harpy scales is pretty tenuous). The current net is 15000 states and covers a 1000 word vocabulary with a grammar that has an average branching factor[3] of 10, extending over a depth of about 2 seconds (6 words), yielding a grammar net of 1800 nodes. To compute these quantities in general, we must combine a characterization of the grammar in terms of words with a characterization of the words in terms of phones.

At the word level a language may be characterized as having a vocabulary of V words with a redundancy of r, ie, the fraction r of the words in any sentence that may be deleted without loss. The redundancy characterizes the aggregated constraint from the grammar and static semantic restrictions. Then, for an utterance of length Dg words the number of nodes in the grammar tree, Tg, can be expressed as:

$$(3.6)\quad Tg = Bg^{Dg} = V^{(1-r)Dg} = (V^{(1-r)})^{Dg}$$

The grammar net arises from this tree by identifying many nodes. This can be thought of as compressing the branching factor by an additional factor of c in determining the growth rate of the grammar net. In Harpy the grammar tree is 10^6 (for 6 words) and the net has only 1800 nodes, yielding a total compression of about 550, or a compression factor per word of c = .35. Note that this does not represent an increase in the redundancy of the

3 There is much variation and some inconsistency in branching factor estimates. They arise in part because of the difference between taking arithmetic and geometric means and in part from the actual paths taken through grammars differing statistically from random paths.

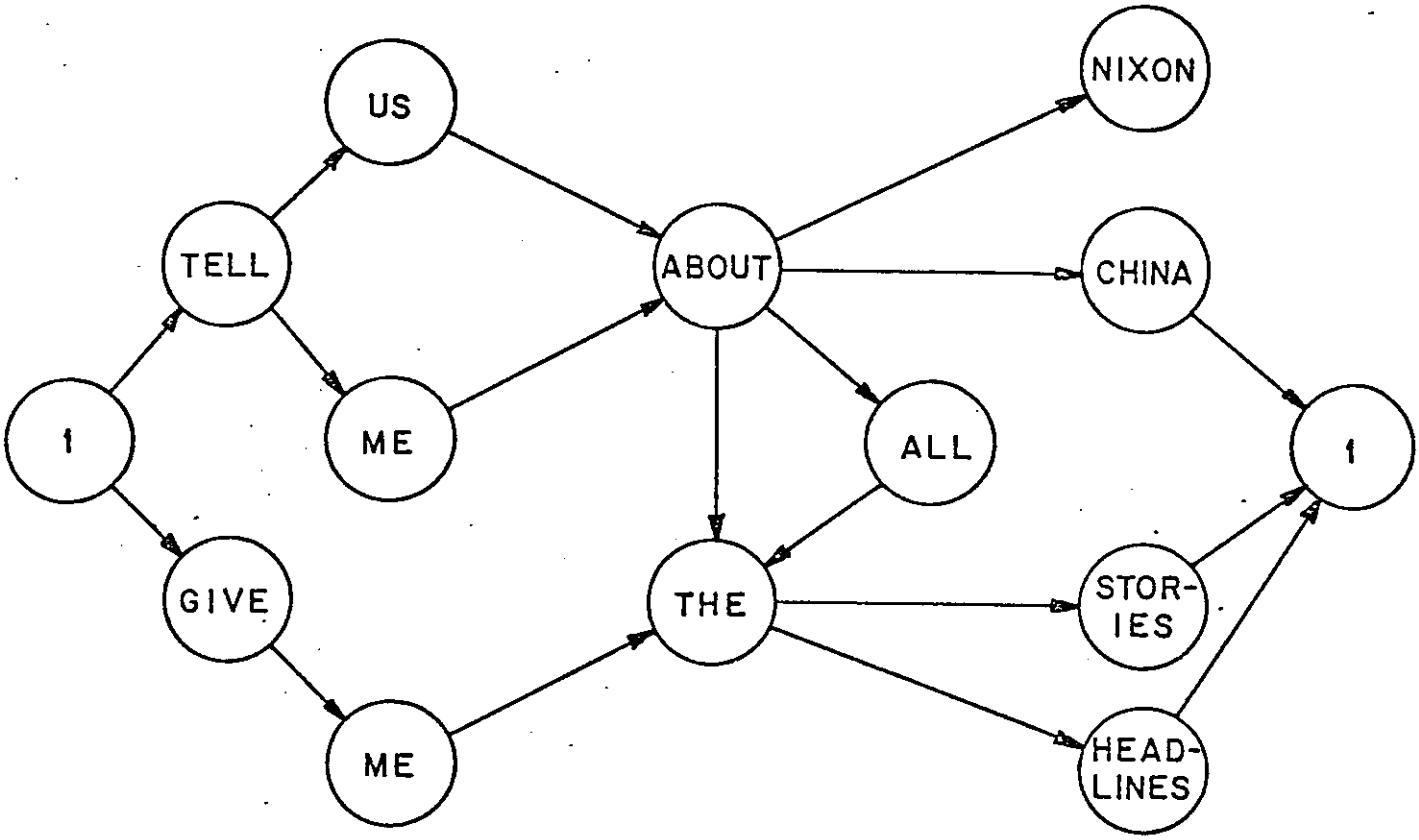


Figure 5. Fragment of grammar net

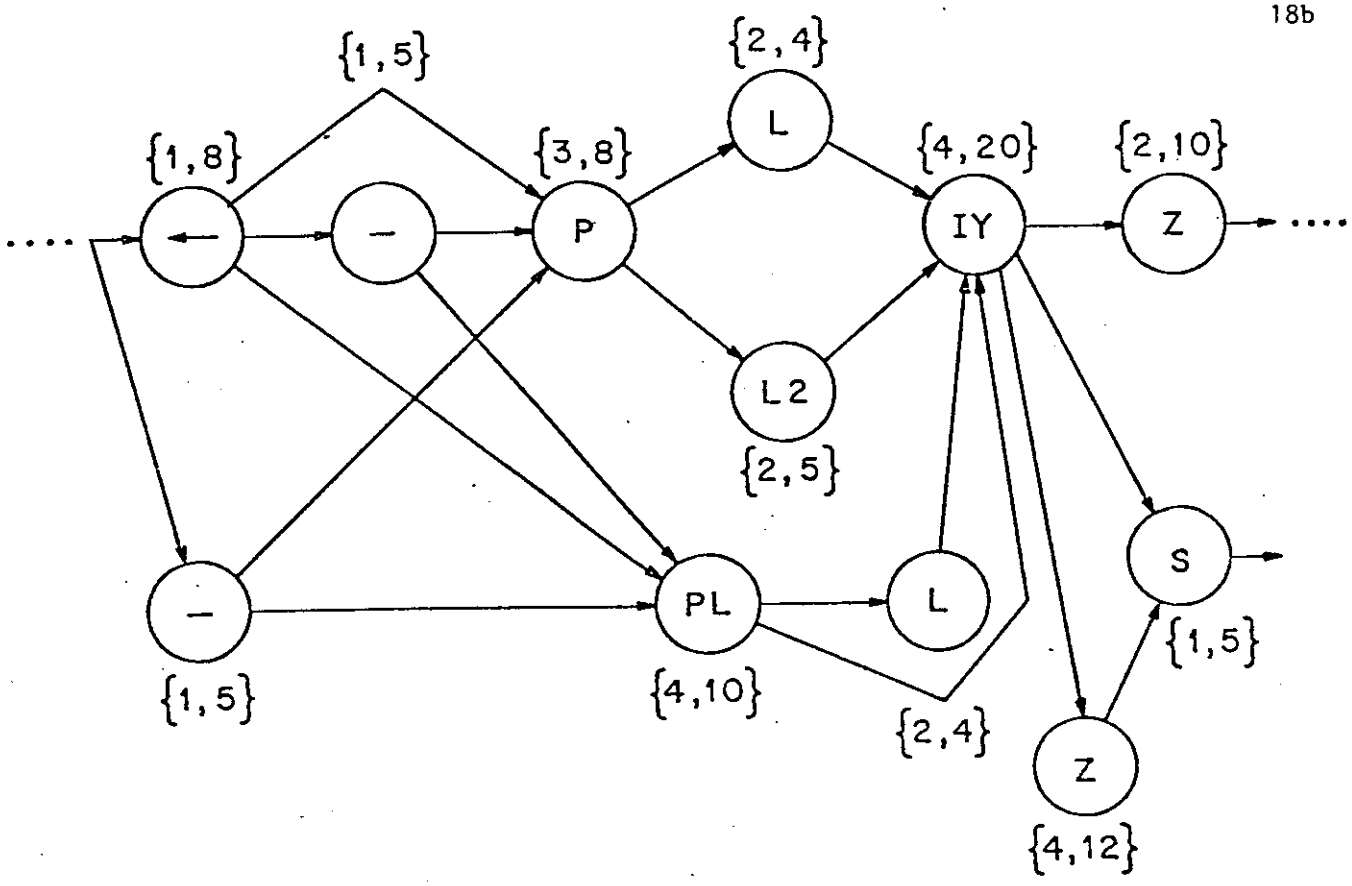


Figure 6. Example of phonetic network of the word "please"

(S1) Can Harpy recognize speech?

(S2) Can Harpy be extended to full speech?

(S3) Can Harpy recognize full speech in time?

(S4) Does Harpy require too much immediate memory?

(S5) How does Harpy cope with the creativity and variability of speech?

(S6) Can Harpy respond adequately to speaker variability?

(S7) Is language acquisition possible with Harpy?

Figure 7. Sufficiency analysis issues for Harpy

language; the effective branching factor is still V^{1-r}. No one knows how c extrapolates to a full language. If, from ignorance, we take c as a constant, then for the states in the grammar net we get:

$$(3.7)\quad Sg = (cV^{(1-r)})Dg$$

The total states in the Harpy net arise by replacing each node of the grammar net by a lexical net, which both expands the states and lengthens the depth. To first order the average lexical net is independent of vocabulary size and grammatical structure (though words do get longer as vocabulary increases). There are Sw states per word; thus we get for the total states in the Harpy net:

$$(3.8)\quad S = SwSg$$

The average branching factor, B, of this net (the average size of the transition set T) arises from the branching factor within words, Bw, and the branching factor of the grammar, Bg, which is $V^{(1-r)}$ from (3.6). The latter takes place only over word durations, not over each state transition. Since the words are on the average Dw transitions long, we get for the effective branchiness (ie, transitions):

$$(3.9)\quad B = BwBg^{1/Dw} = BwV^{(1-r)/Dw}$$

These formulas characterize the language by the vocabulary, V, the redundancy, r, the relevant sentence depth, Dg, and the data for words in terms of phones: Bw, Dw and Sw. The phone representation of a word can be taken as roughly constant, with the values Dw = 5 phones per word, Sw = 8 states, and Bw = 2, all taken from Harpy. Dg can also be taken as constant, since it represents not the average length of a grammatical utterance, but the maximum length of phrase that carries some significant constraint. Dg = 6 (3 words/sec * 2 sec) is consonant with Harpy. For vocabulary and redundancy we can adopt ranges of values. A full vocabulary lies between 10^4 and 10^5. The redundancy of regular English lies between .5 and .75, as derived in various ways (Luce, 1960). Vocabulary and redundancy always enter into the calculations as $V^{(1-r)}$, the effective grammar branching factor (Bg). Therefore we need only consider values of Bg: Bg = 10 to 320 spans the full range. In Figure 8, we show the various combinations of V and r that combine to produce a given Bg.

Figure 8 shows the calculations for S and B as a function of Bg. The first column is for current Harpy, the others are the extrapolations. We see that S ranges from the order of 10^3 states, for a vocabulary of 10^4 words and a redundancy of .75, to a horrendous 10^{13} states, for a vocabulary of 10^5 and a redundancy of only .5. A million states does not perhaps seem out of the question, but it is clear that the the number of states can go out of sight rather easily. By contrast, we see that there is relative small variation in the figures for branching factor, from 3 to 6.

These calculations do not doom a Harpy-like scheme. Additional ideas about how the number of states can be reduced are not reflected here. These calculations do tell us to watch the total numbers as we consider how Harpy is mapped into human cognition.

(S3) Can Harpy recognize full speech in time? The computational task of Harpy is easily expressed given the basic parameters. The amount of computation is proportional to the states considered (ie, the beam, F) times the branching factor (B) times the number of steps (D). Each unit of computation involves a comparison of a template to the data, a step in the logic of maximization and selection, and an accessing of a transition and a new state. It is best expressed in amount of computation per second of speech:

(3.10) Computation = F*B*D

The effort to do the backtrace is negligible compared to computing the forward beam, so is not included in (3.10).

D is fixed for a second of speech at about 5*3 = 15 steps per second. We know the beam cutoff fraction (f) for the existing Harpy (.01) but do not know how it would extrapolate. What matters is how good the acoustic separation is and how many candidates there are. The criterion keeps all those candidates within a band at the top. Assuming this fraction remains constant for full speech lets us plot on Figure 8 the various amounts of computation for variations in S and B. This yields a minimum amount of computation per second of 10^4 units, according to Figure 8. This is an amount of computation, not the time to make the computation, since it remains open how big the unit is and how much concurrency might occur. There are limits to the amount of concurrency possible, but that depends on the computational scheme.

(S4) Does Harpy require too much immediate memory? The Harpy algorithm demands that a set of states, the beam, be carried through time. The beam is about 200 states for current Harpy. Assuming the beam fraction to remain the same at .01 yields the estimates shown in Figure 8. At the least a few hundred states seems to be needed. This implies a requirement for immediate memory in the human. Of course, this need not be standard verbal short term memory (the seven plus or minus two variety), but the requirement may still be hard to meet.

The requirement is actually more than the momentary set of states in the beam, since the past beam must be remembered (in terms of back-pointers), so as to recover the optimal path for the final recognition. Thus the immediate memory requirement is probably more like the total integrated beam, which is F*D, where D is the total steps, namely 30. These figures are also shown in Figure 8.

The requirement for immediate memory is not just for space, as discussed above, but also for duration. The backtrace does not commence until the forward sweep has covered the whole utterance. This is an essential feature of the algorithm, since it is the way in which the results of the search is obtained. In Harpy this duration is of the order of two seconds. In any event, this duration is an additional requirement on the nature of immediate memory.

More subtle requirements on memory organization arise from the iterative nature of speech. The acoustic data arrives continuously over time at exactly the same receptor

Vocabulary	Redundancy					
10↑5	.80	.74	.68	.62	.56	.50
5*10↑4	.79	.72	.66	.60	.53	.47
10↑4	.75	.67	.60	.52	.45	.38
5*10↑3	.73	.65	.57	.49	.41	.32
10↑3	.67	.57	.47	.37	.27	.17
Grammar Branching (Bg)	10	20	40	80	160	320
Grammar States (Sg)	2*10↑3	1*10↑5	8*10↑6	5*10↑8	3*10↑10	2*10↑12
Harpy States (S)	1*10↑4	9*10↑5	6*10↑7	4*10↑9	2*10↑11	1*10↑13
Branching factor (B)	3.2	3.5	4.2	4.8	5.5	6.3
Computation units	7*10↑3	5*10↑5	4*10↑7	3*10↑9	2*10↑11	1*10↑13
Beam (F)	2*10↑2	9*10↑3	6*10↑5	4*10↑7	2*10↑9	2*10↑11
Integrated Beam (Fx)	4*10↑3	3*10↑5	2*10↑7	1*10↑9	7*10↑10	5*10↑12
Fixed parameters:	Bw =2	Dw = 5	Sw = 8	Dg = 6	c = .35	f = .01

Figure 8. Estimates of critical parameters for Harpy

organs. Thus, ipso facto, the signal must move to other organs in order to be remembered (or, equivalently, to make way for the new signal). This suggests pipeline computation. But a pipeline implies that computational capacity exist at each stage of the pipeline, and may imply a large duplication of computational mechanism. Actually, current implementations of Harpy do not pipeline. Having only a single processor, they pass the data through the processor and then lay it out in (memory) space to be retrieved later for the backtrace. The active computation is really an instantaneous effort on the front of the speech wave, and all of the past remains quiescent until the backtrace. Thus the memory organization problem is really how to get the memory for backpointers moved out so that it does not clog up the current computation. The duration of processing before backtrace is not likely to be constant; rather, it would seem to depend on the actual dependencies in the speech. This variability may also be a complication in dealing with the memory organization.

(S5) How does Harpy cope with the creativity and variability of speech? Harpy works with a fixed network. Hence an immediate concern is whether Harpy by its nature is inadequate to understanding freely expressed speech. There leaps to mind the long standing arguments of the linguists about the creativity of language. Along with this come questions of fragmentary and ungrammatical utterances.

There are real issues here, but one must first lay to rest the proposition that finiteness always brings with it a failure of creativity. Linguistic creativity has never seemed impossible because the alphabet is finite or even because the vocabulary is finite. It has always been understood that iteration of finite means (here, selection from a finite vocabulary) is potentially sufficient to deal with creative expression (here, creative reception). Harpy can be viewed merely as an *extended vocabulary* that takes into account the redundant structure necessary to extract speech from the noisy signal (not to understand its semantics), so that the finally recognized utterance is an iteration of trips through the net. Thus, it is no more (and no less) true of Harpy than of any system that the set of utterances is limited to a finite (hence "non-creative") set.

The real issues of creativity and variability seem best dealt with as requirements to handle certain specific situations. First, a human can understand grammatical utterances that he has not heard before. Does Harpy depend on the net encompassing all grammatical possibilities? Does the view above of Harpy as an extended vocabulary plus the use of very large nets suffice? Second, a human can understand any sequence of words from its vocabulary, if clearly spoken. Such utterances surely lie outside the Harpy net. How can Harpy recognize such utterances? Third, humans cope with some utterances that have genuinely new elements, exhibiting thereby various forms of cognitive assimilation and inference. How can Harpy do likewise? Fourth, humans are able to incorporate immediate increments of linguistic knowledge; eg, a new word or notation may be explicitly defined en passant during conversation. Such knowledge cannot be in the existing network when given. Currently Harpy needs a formal representation of the grammar and a substantial compilation phase. How can it handle such short term knowledge? Fifth, humans cope with some grammatical errors and fragmentary utterances. These must also surely lie outside the Harpy net. How will Harpy cope with such "structural" errors, as opposed to continuous errors in the acoustics (which the template match using the Itakura metric at least addresses)?

These requirements may be solved by a variety of different mechanisms. We group them together since they all involve somehow the fixed Harpy network and how it is to be transcended.

(S6) Can Harpy respond adequately to speaker variability? There should be a sufficiency issue surrounding the Harpy mechanisms for expressing speaker variability by means of unique phone sets for each speaker. The Harpy scheme does not require much memory, so long as there are a fixed set of phone labels and all that varies are the templates. But there are time lags on obtaining the variability and presumably there are limits to how much variability can be handled by this mechanism. Though Harpy shows some adaptiveness for individual speaker's phonetics, it has no mechanisms to deal with individual differences in phonological and grammatical rules.

(S7) Is language acquisition possible with Harpy? Any sort of elaborate structure or complex preparation raises the issue of how such a structure is acquired. Though the homogeneous structure of the network, made up of simple elements (nodes, transition links, templates), seems relatively non-problematical a priori, the elaborate compilation process raises a red flag. The computational aspects of that process must be taken into account. Even more, the acquisition of the compilation process must be accounted for. This becomes an especially binding issue for the human system, since it acquires language very early, when its cognitive powers are apparently still primitive. The natural assumption is that the basic acquisition of language must be a feature of the underlying cognitive system and not the result of extended intelligent strategic action.

For the current Harpy, the compilation process involves having additional representations for all the components of language: a grammar, a lexicon, and acoustic-phonetic rules for word junctions. Actually a lexical item for Harpy is already an integrated representation of all the different ways in which the word can be pronounced. Conceivably this knowledge is held in rule form, so that a new word can be expanded into a net by some generation process, just as is the grammar net. The alternative is that knowledge about variability of pronunciation arrives in piecemeal fashion, so that the net representation of a lexical item must either be regenerated or modified. The same is true of the grammar net, of course. Though the acquisition of additional grammatical constructions may be rare, each lexical item added to the vocabulary requires an extension of the grammar net (as well as the total compiled network).

Summary We have enumerated some of the main sufficiency questions. These arise directly from the structure of the Harpy algorithm. They need to be treated no matter how we map Harpy into the human cognitive system. However, they are not the only questions that will occur. Once we consider a specific identification, additional questions may demand treatment, depending on the shape of the mechanisms that are employed to realize the Harpy algorithms.

It should not be inferred that all these problems are Harpy's alone. Analogs of some of them will exist for other schemes. For instance, one of the underlying objections to

Analysis by Synthesis has always been a sufficiency issue of how there could possibly be enough computational capacity for it to run in a blind generate-and-test mode, even if it did have a good test (the synthesis).[4]

The questions of creativity, and the amount and duration of immediate memory, will arise in all cases. In fact, one of the gains to be made from attempting a sufficiency analysis of Harpy is just to get these issues to claim their proper share of attention in competition with various empirically striking phenomena.

We have taken Harpy as it currently is. There are many things still to be done to Harpy to decrease its computational complexity (both time and space), increase its accuracy and increase its flexibility (both in performance and in acquisition). These considerations can branch off in many directions. It is not worth following them until they become critical to the analysis. Thus, we will come back to possible modifications of the Harpy structure after we have attempted to map Harpy into a model of human perception and have specific difficulties that demand solution.

4 It never proved persuasive that Halle and Stevens (1962) also posited a plausible generator, since it was a genuine *deus ex machina*.

4. HARPY AS A PRODUCTION SYSTEM: PRELIMINARY ANALYSIS

We now turn to the central task of the paper: to find a way for the human to perceive speech by performing a Harpy-like computation. The key to doing this is to take Harpy seriously and not worry too much about outrageous aspects of the scheme until the mapping is complete. Then we can ask whether the difficulties so revealed can be alleviated. Thus we have termed this section a preliminary analysis.

We assume HPSA77, the architecture described in Section 2, which shares the properties of a basic production system architecture and adds four (complex) assumptions, D1-D4. Not all these assumptions are equally critical for us. Having developed the architecture independently, so it qualitatively fits a modest range of behavior, it is appropriate to posit the entire architecture. However, nothing is sacred about HPSA77. If the requirements of Harpy point in a different direction, we will follow Harpy and modify HPSA77.

4.1. The Essential Idea: Transitions are Productions

The basic elements of the mapping can be immediately set out:

(M1) The states in Harpy correspond to symbols or data elements.

(M2) The state-dependent transitions correspond to productions with prior states in their conditions and successor states in their actions.

(M3) Most transitions correspond to variable-free productions.

(M4) The parametric representation of the acoustic signal arises in WM as a data element.

(M5) The comparison of templates to data elements occurs by means of the recognition match.

There is no automatic guarantee that a mapping with these gross properties can actually be constructed. There are many open issues: Can variable-free productions represent the computation? Can the recognition match perform the required act of comparison? How are the acoustic parameters actually to be represented? And so on. Still, some important things about sufficiency follow from these five mapping propositions. It is worthwhile noting them before plunging into the detailed considerations of the full mapping.

The mapping localizes the speech perception productions where HPSA77 independently claims perceptual processes belong, namely, in the productions corresponding to Shiffrin and Schneider's automatic processing. One important consequence is that

concurrent processing is available, something that is clearly needed. The fundamental act of firing a production will correspond to the basic computational act of making a transition in Harpy. Thus the beam will be sitting in WM and will consist of elements holding states and the values associated with the state. Concerns about the size of the beam then translate directly into concerns about WM size and duration. The speed of computation becomes tied directly to the cycle time of the recognize-act cycle. Note that in HPSA77 the cycle time is not yet pinned down, either for automatic or controlled productions. (Assumption D1 dealt only with avoiding the implication from the Sternberg coefficient of too short a cycle time.)

An important feature of the mapping is that it breaks the network apart. What exists is a set of individual productions. They come together dynamically to produce the Harpy computation. The act of construction of the net is the act of adding many individual productions. Thus, it appears unnecessary to engage in a major act of compilation. Too much comfort should not be taken from this, since many aspects of the acquisition process remain problematical, such as creating appropriate individual productions, modifying or screening the behavior of inappropriate productions that have already been learned, etc. Still the problem is cast in a somewhat different computational light.

A last general implication is that the perception of speech does not take place in a separate component from the rest of cognitive processing. Though of necessity most processing must be done by variable-free productions, variable-containing productions can also participate to the extent they can be executed in limited time. Thus a way exists for modest amounts of general computation to be brought to bear during perception.

With this overview, we can proceed to carry out the mapping in detail.

4.2. The Detailed Mapping

Many issues must be dealt with in making the mapping. We will introduce them one by one in an attempt to make clear what aspects of the model are responsive to what issues. We will sometimes cope only partially with an issue. By laying them out concretely it will be possible to accumulate the set of partially satisfied issues at the end of this preliminary pass.

It is possible to foresee what will drive the mapping. Harpy requires a lot of computation, much more than can be accomplished serially in the time available. The only device in HPSA77 for concurrent computation is the execution of distinct variable-free productions. Thus it is necessary to incorporate as much of the computation as possible into the recognition match. Getting a large number of elements in working memory for a few (powerful) productions to iterate over cannot work. Thus, even when we do not know exactly the form that the full computation can take, we know enough to try to express the computational functions in the productions themselves.

(I1) Performing state dependent transitions. We can try a single production for each transition. Equation (3.3) suggests that we write:

(4.1) $P_{i,j}$: (State S_i) --> (State Sj) For each i and each j in T_i

The WM contains elements for the states (State S_i). We have tagged the elements with an identifier (State ...), though it is not clear it is needed. A transition to a new state occurs if a current state element (State S_i) is in WM. Thus the beam resides in WM as a set of active state elements.

(I2) Incorporating the likelihoods into the transition. Formulation (4.1) is inadequate, since all that S_i encodes is the unique state identifier, and not the accumulated likelihood value, $L_{i,t}$. On the other hand, the likelihoods themselves are just values of some sort and are not identified per se with their states. We can modify (4.1) by making the data element for a state hold both the state identification and the cumulative likelihood value, (S_i L):

(4.2) $P_{i,j}$: (State S_i =L) --> (State S_j [=L)

By encoding L and S in the same element we maintain the association between them. However, (4.2) leaves open how the new L is determined, since its value will generally be different from the original likelihood of S_i which is bound to the variable =L, though partially dependent on it. We indicate this unfinished character of the mapping by the use of brackets, [=L]. Note that we have introduced a variable into (4.2) (=L) to pick up the likelihood from WM. Thus, our system is no longer variable free. Before taking care of this difficulty, we need to get the rest of the path computation expressed. Note also that (4.2) leaves open how likelihoods are represented, specifying only that the representation can be bound to a variable.

(I3) Taking account of the current acoustic data. The acoustic data enters into obtaining the new likelihoods and we can express this dependency symbolically within the brackets:

(4.3) $P_{i,j}$: (State S_i =L) (Data =D) --> (State S_j [=L + $C(A_i,$=D)])

The WM now contains an element that holds the acoustic parameters, (Data D_t). As with the state elements, we have tagged the acoustic data with an identifier, (Data ...), though its necessity is not clear. In order to let the acoustic data enter into the determination of the new likelihood on the right hand side, we have had to introduce another variable (=D) into the system, which must ultimately be removed. Again, we have left open the representation of the acoustic data (D_t), assuming only that it can be bound to a variable (=D).

(I4) Encoding the backpointers to permit the backtrace. Backpointers can be encoded by remembering the state pairs of each transition that is generated. This can be done just by expanding the state element once more.

(4.4) $P_{i,j}$: (State S_i = =L) (Data =D) --> (State S_j S_i [=L + $C(A_i,$=D)])

The = in the initial template (State S_i = =L) is a don't-care mark, which indicates that the

prior state is not relevant to whether a match occurs. It need not count as a variable, in terms of variable-free productions, since no value is bound and used on the right hand side. However, for this to be the case does imply a further specification of the UV mechanism, namely, that the constraint is on binding or using and not just on multiple instantiation.

This formulation implies that backtracing will occur by pairwise linking back through the state elements, though we leave open for the moment just what calculation has to be performed. This proposed mechanism has a potential flaw. If a state were ever repeated then the back pointer would not be unique; either a loop or a jump might occur. In current Harpy this is avoided because a backpointer array is kept for each time step. But there is as yet no explicit time indexing in the WM representation, so confusion is possible.

The PS described by (4.4) will carry out the generative component of the forward sweep. The set of states that are developing in the WM at a given instant constitutes the beam at that instant. But the WM holds the entire beam history, with each new wave of the beam showing up at the front of WM (ie, as the most recent set of state elements). The productions must still become variable-free, so that as many as are satisfied can fire; thus, the entire beam can be computed at the time its acoustic data element arises in WM.

The system so far has only been responsive to generating the alternatives. We now begin to put in the selective mechanisms.

(I5) Taking the maximum over all states in the path equation. Taking the maximum is one way of cutting down the set of paths to be carried forward. Consider not computing at all the maximum called for in equation (3.3). The result would be to carry forward several paths to the same state. These would then have duplicate histories for the remainder of the utterance. They would expand the total set of states that are candidates for retention in the beam. Cutting back the beam to fixed size would presumably get rid of some of them, but the net effect would still be to cause some fraction of the beam to consist of useless paths. Providing that the maximum state would be taken at the end and that one could correctly trace out the back path, the only result would be an effective diminution of the beam. Stated otherwise, if the fraction lost was stable, then the maximum could be dispensed with if compensated by a beam of increased width with its consequent extra computing costs.[5] Thus, one possible solution for taking the maximum is to ignore it.

However, it is possible that we need not follow this course. The recognition match takes the maximum automatically, under the proper mapping. To see this, consider in detail what happens when two states, say S_i and S_j, lead to the same state, S_k. Each then will lead on to duplicate common paths, which will differ only in the likelihoods that are brought forward, one reflecting the history through S_i, the other the history through S_j. We can plot the state elements that will be generated in WM at the critical junction, moving on to (say) S_m and then S_n:

5 This holds only for systems, such as Harpy, that use a beam search, where beam clipping performs the same function (up to some point) as taking the maximum.

(4.5) (State S_k S_i L_i) => (State S_m S_k L'_i) => (State S_n S_m L''_i) => ...

(4.6) (State S_k S_j L_j) => (State S_m S_k L'_j) => (State S_n S_m L''_j) => ...

Suppose L_i were less than L_j (so that, with maximization, only the S_j-S_k-S_m-S_n path would survive). Then L'_i will be less than L'_j, L''_i less than L''_j, and so on through the future. Now the generation productions will certainly generate both of the state elements at the left side, coming into state S_k. They may generate both of the middle elements in (4.5) and (4.6), because their state elements for the conditions are still dissimilar (one has S_i, the other S_j, in the second place). Furthermore, the output elements (the middle elements) are also dissimilar, one having L'_i, the other L'_j; so they will not coalesce into the same WM element. But will they generate both of the elements at the far right? These are generated by (4.4) operating on the two middle state elements, respectively. These input elements differ only in L'_i and L'_j and are both being matched to the same production (since the acoustic data is the same for both):

(4.7) $P_{m,n}$: (State S_m = =L) (Data =D) --> (State S_n S_m [=L + $C(A_m$,=D)])

Could conflict resolution accept one (with =L bound to L'_j) in preference to the other (with =L bound to L'_i)? If a preference does occur, then it is likely that the secondary item will never get to fire. Both recency and special case order are candidates to accomplish this. If the difference between L'_j and L'_i is to be one of recency, this implies something strong about the encoding of likelihoods, namely, that it be analogous to activation. If the difference is to be one of special case order, then no such far reaching implication seems to exist. Let us hold this choice in abeyance for the moment, but assume that some mapping decision will be made so that only one of the two instantiations fires. Then the potential dilution of the beam with duplicates will come to a halt almost immediately after the convergence of states. Possibly some tag ends will exists (the middle element of (4.5)), but this will not prove bothersome. We will have to return to make this decision in final form, but can leave it for the moment.

(I6) Clipping the beam to keep a specified beam width. From equation (3.4) the clipping conditions involve rejecting all those state elements (State S_i S_j L) where L is more than e from the best. The cutoff varies so as to maintain a given robustness in the beam. Clipping can be achieved by two alternative mechanisms: (1) failing to fire a production on data elements that do not warrant it (ie, should be clipped); or (2) recognizing unsatisfactory elements in the WM by general clipping productions that mark the elements to inhibit further generation. General clipping productions would probably contain variables. As noted earlier, it is not possible in general to use such productions to evaluate state elements, since they would not be able to fire in parallel. Thus, let us explore the other alternative, which is to incorporate as much of the evalution as possible into production selection.

Consider modifying production (4.4). First, the likelihood variable (=L) can be replaced by a match to a constant, so that degrees of matching can be introduced. Second, C, the comparison operation, can be moved to the condition side; thus =D can be replaced by the template and the recognition match itself be used to make the comparison. We get:

(4.8) $P_{i,j}$: (State S_i = L) (Data A_i) --> (State S_j S_i [L - r(L) + r(A_i)])

Where r(X) = the residual from matching to X

Two important assumptions have been made in (4.8). To explain them, consider that (4.8) was matched against:

(4.9) WM: (... (State S_i S_k L') ... (Data D_t) ...)

Normally, in HPSA77, a match will occur if and only if L and L' are the same, and similarly if A_i and D_t are the same. The first new assumption is that L and L' will match if they are "close" and, similarly, A_i and D_t will match if they are "close". What constitutes "close" remains open, except that ultimately the closeness of A_i and D_t must agree with the template metric used in the Harpy algorithm (though clearly it need not be the Itakura metric, used in the current Harpy, as others may be equally well suited). The second assumption is that the residual of the match of a condition, call it r(X), is available on the action side. The brackets on the right hand side of (4.8) remind us that we have not settled how the computation there will go; but both residuals show up there as appropriate input to determining the new likelihood.

These assumptions indeed appear radical. However, Harpy works with continuous quantities, and the incorporation of quantities into a symbolic system, such as HPSA77, must occur someplace. The scheme implicit above may not be right. There are alternative paths that might be traveled. The most obvious one is to retain the exact match and to introduce multiple productions with quantitative symbols that range over discrete levels of likelihood. The chief drawback to this type of scheme, even beyond the explosion of productions, is its limited precision. Thus, we will pursue the more radical path. However, as with the assumption about conflict resolution for taking the maximum, we will accept the result and proceed with the remainder of the mapping. When all assumptions have been made we will be in a better position to analyze their joint nature and impact.

The same likelihood symbol, L, appears in all productions of (4.8). This amounts to adopting a uniform basis for the measurement of likelihoods. Conceivably, this could be biased for different transitions by using $L_{i,j}$ instead of L. But it is unclear whether this is useful.

One result of this formulation is the apparent elimination of the variables in system (4.4), thus returning to variable-free productions that can run concurrently. Whether this is effective depends on how the residuals are handled. They clearly sense something of the elements matched and they may turn out to be variables in sheep's clothing. Note that L - r(L) is just L', which is to say we have substituted L' for L. How is this different from simply matching to =L and substituting? In any event, this cannot be evaluated until we finally settle how the computation in the brackets is done.

(I7) Extrapolating speech over gaps in the input. An important problem should be

noted in (4.8). (Data A_i) can fail to match independent of the satisfaction of the accumulated likelihood, however it will ultimately be sensed. This produces a situation of sudden death in which a single unfortunate gap in the acoustic data can terminate the optimal path. This gap need not be caused by internal noise in the system, but for example might be from external noise (eg, a door slam). This problem is explicitly recognized in the current Harpy system, where the criteria on the cutoff is often expressed in terms of how many cycles it permits a path to survive when it is receiving no support from the acoustic evidence at all.

Let us return to the idea of selecting the beam after the forward generation has developed the new state elements in WM. Then generation need not be dependent on the acoustic data. However, we must be sure that selective evaluation is done by variable-free productions. This must yield two sets of productions, one for generation, one for evaluation:

(4.10) $P_{i,j}$: (State S_i = L) --> (State S_j S_i [L, r(L)])

(4.11) $Q_{i,j}$: (State S_j S_i L) (Data A_i) --> (State S_j S_i [L - r(L) + r(A_i)])

If L is close enough to L' (the likelihood in the state element), productions (4.10) will take every member of the beam and make all the transitions, thus yielding the full expanded beam. We have left open (as far as possible) what value is actually produced for the likelihood of the new element. This is indicated by the brackets. Once a state element is in WM, productions (4.11) can be evoked if both L is close enough to L' and A_i is close enough to D_t. Here the brackets on the right side express the resultant in the same form as (4.8).

Even if the acoustic data is too far from A_i for (4.11) to match, then (4.10) will continue to carry the expansion forward. If at some subsequent point the match becomes good enough, then (4.11) can again be evoked to do whatever the brackets indicate to raise the value of the likelihood.

As it stands, (4.11) produces extra elements which differ only by the value of the likelihoods. We leave open whether this causes any difficulty and whether the issue can be avoided by interpreting (4.11) to modify an element rather than replace it.

The obvious difficulty with the system (4.10)-(4.11) is that (4.10) would appear to generate forever, unconstrained. This means that the bracket computation must do something that diminishes the likelihood of the new state elements, so that if unreinforced they will eventually fall too far below L to be satisfied. The obvious way to imagine this happening is for the value of the likelihood to be common through all these forward steps and to gradually decay if left alone. The obvious production to replace (4.10) would be:

(4.12) $P_{i,j}$: (State S_i = =L) --> (State S_j S_i =L)

The value of =L decays autonomously if left alone.

It is clear that (4.12) has re-introduced variables -- just what seems prohibited. Replacing =L with [L - r(L)], which produces the same value as a reduced L' would seem to be just a notational difference.

All is not yet lost. Consider the following scheme (which will add to the accumulating list of promissory notes if adopted). We ignore the residual in (4.10), but do use a diminished value of stipulated likelihood. Thus we get:

(4.13) $P_{i,j}$: (State S_i = L) --> (State S_j S_i J_i)

J_i = The stipulated diminished likelihood

If $P_{i,j}$ matches at all, then the prior value of the likelihood is lost, to be replaced by a stipulated value (J_i). Now we assume that the chance of $P_{i,j}$ being satisfied does depend on the closeness of the match. Thus there is some probability p(r(L)) that (4.13) will fire and this diminishes with increasing r(L). If J_i is set so that the probability is p, then the probability of a particular forward path surviving undergoes exponential decay, ie, 1, p, p^2, p^3 ... for successive steps forward. If p times B, the branchiness of the transitions, is less than one, then the entire expansion from a state element will converge to zero as long as there is no (positive) interference from the acoustic signal. Note that p can be made to vary as a function of the state (ie, J_i is a function of state), so that a priori high probability paths can be distinguished from rare paths.

There are some uncomfortable features with this solution. One is that the choice is actually probabilistic so that the best path has some absolute chance each time of disappearing. Furthermore, attempts to attain high survival rates translate into long survival times.[6]

A second even more uncomfortable feature is that all history is wiped out as each new generation is selected. L', the cumulated likelihood of the state up to this moment, does not pass to the new state elements. L' affects only the chance of survival on the one trial. This might conceivably be tolerable when all the forward branches are data-free extrapolations across a complete acoustic gap. But this same production will be used for all generation, and so will have the same property of losing the past. Despite these difficulties, let us hold this solution as one possibility, along with the chance of finding some way of retaining the accumulated likelihood value as the system moves forward.

Before leaving this issue we should note one more potential difficulty to the whole notion of splitting generation and evaluation into separate productions. It is possible for the productions to get out of synchronization. Although a (4.11) production cannot operate until its corresponding (4.13) production has fired, there is no such absolute constraint in the

6 This problem also plagued the fixed (10ms) step-size version of Harpy in implementing variable length segments entirely through self-looping transitions. For then segment length must be geometrically distributed, for exactly the same reason as survival durations above.

other direction and generation might get ahead of evaluation in some uncomfortable way. Given that all productions are executing concurrently, there may be no difficulty, but the potential hazard sets a red flag.

(I8) Adding likelihoods to comparisons. If we accept productions (4.13) and (4.11) as an adequate representation of the recognition system, then the problem of combining continuous quantities is focussed in (4.11), and in particular in the bracket computation [L, r(L), r(Ai)] plus what can be transmitted from the condition of a production to the actions without evoking the UV mechanism. We have already come across the difficulty in dealing with residuals that they appear to be variables after all.

Productions (4.11) turns out to be potentially a special case where the UV mechanism might be avoided, though we may not wish to exploit it. Namely, the action expression is identical to the condition expression except for the change in value of the likelihood. If we view the action element as new, then there appears no way to avoid using a variable to communicate the information across the production. But if we view the action element as being the same element as the condition element then the action is only one of modifying what is there. The possibility of conceiving it this way depends on how likelihoods are encoded. If the updated likelihood is a new arbitrary symbol, then again there seems no way out of treating it as a substitution. If the updated likelihood is truly a changing of value, then a different operation is conceivable. It is worth noting that this possibility seems less open to us in dealing with (4.10). The action element (State S_j S_i [L, r(L)]) is a new element in WM and hence does not pre-exist to be modified. This difference is not fortuitous, but follows directly from splitting (4.8) to separate the creative-generative aspect from the evaluative aspect.

If we take the path that lets us map Harpy successfully, then we can invent an action, call it *Increment*, which lets us add the residual $r(A_i)$ to L. We need also to indicate that this happens on a given element identified at the input. To remain consistent with our characterization of productions as variable-free, we will prefix the action element with "=". Thus we replace production (4.11) by:

(4.14) $Q_{i,j}$: (State S_j S_i L) (Data A_i) --> =(State S_j S_i Increment($r(A_i)$))

This use of = does not constitute use of a variable, since the identity is being carried by the element as a whole.

Besides placing additional requirements on the encoding of likelihoods, we have also specified more narrowly the critical aspects of the UV mechanism. Assumption D1 does not specify what aspect is really limiting -- the instantiation of variables, the transportation of value, the creation of new elements, or the replacement/substitution of values in these new elements. These different aspects have quite different properties (eg, how processing time varies with the number of variables, variable occurrences in conditions, action elements, and variable occurrences in actions). But it was not necessary to specify the effect so tightly. The present assumption seems to put the burden on the creation of new elements, though it still leaves open various other aspects.

If we use (4.13), a small discrepancy between it and (4.14) needs to be cleared up. If (4.13) diminishes the likelihood from some initial value L, then the state elements that (4.14) will deal with will have likelihood J_i. Consequently, instead of (4.14) we should use:

(4.15) $Q_{i,j}$: (State S_j S_i J_i) (Data A_i) --> =(State S_j S_i Increment(r(A_i)))

This scheme represents adequately the process of constructing cumulative likelihoods either on the original scheme in which such likelihoods would be accumulated step by step or the new scheme in which each step leads to a history-losing resetting of the likelihood to a standard value (the J_i).

(I9) Matching the template and the acoustic parameters. As shown in equation (3.5), the Itakura metric ultimately reduces to a dot product match, in which the terms from the template really correspond to orthogonal components, rather than corresponding components to the acoustic parameters. There is nothing in the type of match used in production system architectures that looks like a weighted sum, especially with negative weights, which is implied by the notion of orthogonal components to a positive vector (the acoustic parameters). Rather than attempt to cope with such a metric, it seems better to deny that the Itakura metric is other than a computational device for current Harpy, and to look for matches that seem more appropriate to the sort of matching structure available. Certainly the experience with Harpy and other speech efforts (Goldberg, 1975) shows that no one metric and/or set of parameters is uniquely suited for recognition. There seems to be general agreement on the non-essential nature of the Itakura metric. For instance, current research on Harpy involves consideration of new metrics (see also Klatt, 1977).

Having said this, little more can be added. The template A_i in (4.15), of course, is not a single symbol (even with an associated quantity). A_i consists of an expression with symbols for each parameter:

(4.16) (Data A_1 A_2 A_3 -- A_{14})

It is almost entirely open in the system what happens when two such expressions are matched, ie, what sort of summary quantity emerges. We are committed that this summary, which is the residual r(A_i), is the desired metric, and that it can be recovered to be used in the Increment operation.

We conclude that there is no obvious obstacle with respect to the metric, but that we are probably missing the key facts.

(I10) Avoiding confusion with old data. An important simplification of our Harpy production system is that the data elements are not specifically indexed on time. The back pointers, for instance, are simply associated with states and not with states at specific times, which is how the current Harpy encodes the memory. What keeps PS.Harpy from getting all confused, either going around in circles or applying the acoustic data to states of at different times?

The answer must lie in the recency conflict resolution rules, which keep the attention of the system focussed on the front of WM, and in the refraction rules, which keep productions for repeating themselves exactly. To first order, these mechanisms are constructed exactly to keep processing straight over time and they can be presumed to perform adequately here.

If there is any implication here, it is that these conflict resolution rules have to be extended to variable-free productions and are not just a feature of the UV mechanism. Refraction must apply to variable-free productions. Normally, recency cannot apply to a variable-free production, because any two such productions can fire concurrently, hence without conflict; and a single such production cannot have more than a single instantiation. However, if we permit don't-care elements (=) in the condition, then a single variable-free production can have several potential instantiations. It can only be permitted one, and that one should be determined by recency.

(I11) Performing the backtrace. The backtrace productions depend on the data representation of the back pointers, which has already been fixed in the state element; and on the form of the desired final result, which has not been specified yet. To do the latter would take us beyond the recognition phase, and hence beyond the direct concern of this paper. It is sufficient to show that the appropriate grammatically analyzed word sequence can be recovered. This is just the path through the grammar net with access to the words from the abstract nodes of the grammar.

We need a production system that provides a way to initiate the backtrace, to link back through the states, and to obtain access to the symbols for the words (our surrogate for obtaining a final integrated structure to use in the larger cognitive context).

Harpy initiates the backtrace when the utterance is complete. This is a default design decision based on the (temporary) structuring of Harpy's task environment to work on one utterance at a time. The clue to the end of an utterance is a relatively long period of silence. This maps into the detection of a specific acoustic data element (silence not being the absence of perceived sound, but the perception of a distinct acoustic environment). Thus, we get:

(4.17) U1: (State =S = L) (Data A_s) --> (Utterance S_{end} =S)

This production will be triggered whenever the prolonged-silence data element (Data A_s) occurs during the analysis of an utterance (so that state elements are active in WM). U1 contains a variable (=S) for the state within the state element. According to Harpy, the state element to be picked is the one with the highest likelihood (ie, the maximum of the most recent beam) and this cannot be known in advance.

U1 will instantiate all the state elements in the beam that are close enough to L. Furthermore, nothing restricts U1 to matching only the front wave of the beam; instantiations may come from anywhere. According to assumption D1.2, all of these instantiations should be

executed. This could be a problem, since only the maximum element is to be selected for the backtrace. According to assumption D1.3, there may be a limit on the set actually to be executed, though D1.3 does not specify any details of the restriction. If we are to use U1 to initiate the backtrace, we need to specify these restrictions so that (1) the maximum state element is always evoked and (2) whatever other elements are evoked will not bother the operation of the backtrace. It is plausible that the executed set should always contain the best-matching instantiation, which would solve the first part. This does, however, add to the specifications of D1.3. Though distinct from the assumption to be made in section (I5) to take the maximum, it seems quite similar to it in spirit.

What happens to all the other state elements that might be evoked? Conflict resolution should assure that the maximum likelihood element will dominate in the immediate selection. But the others could cause difficulty as they float in WM. One possibility is that the HPSA77 includes the ability to control the lockout within limits, so only a single instantiation can be kept if desired. Let us put off making a design decision here and simply assume that the only the maximum element need be considered.

The action side of U1 provides an element, (Utterance Sj Si), which is analogous to the state elements, but labeled as the final utterance. It will hold the whole chain back to the beginning. U1 provides the last element it the chain, which is a special state, S_{end}.

The obvious way to trace back through the path is with a single general linkage production:

(4.18) U2: (Utterance = =S) (State =S =S' =) --> (Utterance =S =S')

Unfortunately, U2 is not admissible according to assumption D1.4. It uses the same variable in two conditions, and indeed this identification is the heart of the production. The situation here is exactly analogous to the situation in the Sternberg paradigm which was used as an example in Figure 2. There, A1-A2 would have provided a direct solution to classifying the probe item, if the identity test could have been done on the fly, ie, by the two occurrences of =X using the recognition match. The solution is indicated there by the production system B1-B3, which first creates a testing chunk that brings together the two symbols to be compared. Thus, we get a revision of U2 and add a testing production, U3:

(4.19) U2: (Utterance = =S) (State =S" =S' =) --> (Test =S =S' =S")

(4.20) U3: (Test =S =S' =S') --> (Utterance =S =S')

We have again the problem of many instantiations, since (4.19) is satisfied for all state elements (there being no constraining conditions on its variables). Since (4.20) will select out only the one that moves back a step, there is only the problem of the lockout while all the instantiations of (4.19) are being executed. The irrelevant test elements will not get in the way (providing, as assumed, there is no capacity limits on WM). We can defer whether the lockout problems are severe and take U1-U3 as producing the utterance chain.

The action of U1-U3 will come to an end when the chain runs out. There will be a state element of the form (State S_i S_{start} L), which will lead to an utterance element of the form (Utterance S_i S_{start}), as a distinguished element to start the chain.

Finally, we have to retrieve the words. The states are associated uniquely with nodes in the grammar (though many states lead to the same node). These nodes define uniquely the word associated with the node. Since we are considering a compiled network, we can assume that there is a direct link back from the states to the relevant codes for the words.

(4.21) W_i: (Utterance S_i =) --> (Word Wx)

Wx = The word associated with S_i

The word is not indexed on i (ie, $Word_i$), since many states will home to the same word. Actually, there is no need for every state to associate to a word. Several schemes are possible, eg, either (or both) of the end states have W_i productions, but not the interior states. There is nothing in our present context that lets us design this part of the scheme further, so we simply leave this aspect open. Nothing is affected in the path generation by whether or not W_i productions exist for a given state.

U1-U3 and the W_i constitute the backtrace production system. It has some interesting aspects. First, U1-U3 contain variables. Consequently the backtrace runs as a serial process. It will actually ripple back at the basic cycle time of the system, since each step must be finished before the next one is started. Second, it evokes the words backwards, from the most recent towards the past. This seems on the surface to be a violation of the normal way of experiencing input, ie, forward. Thus there may be an interaction with the design decision D3 in HPSA77, which was responsive to making forward experiencing natural.

(I12) Matching variable duration phones. In Harpy the segments are of variable duration and a template match can be extended to the next segment if it is sufficiently similar. There seems to be little difficulty in achieving this. The acoustic data arises in WM as (Data $D_{1,t}$ $D_{2,t}$ ... $D_{m,t}$), though nothing yet indicates whether it existed in some further decomposed form in WM or whether it arises de novo from sensory mechanisms. Consider the simplest version, where components are fixed (complex) acoustic features, which rise in WM autonomously whenever the feature exists in the acoustic stream. (It remains open how much specialized, but cognitive-context free, processing occurs to develop the feature.) At some instant a set of them will trigger off an acoustic recognition production:

(4.22) R_k: D_1 D_2 ... D_m --> (Data D_1 D_2 ... D_m)

This production has the effect of creating a separate memory of the existence of a set of features, independent of the D_i themselves. It resolves the memory problems mentioned under sufficiency issue (S4), namely that speech by its nature originates in a repeatedly used fixed structure, and some sort of transfer of information to a new place must occur.

The R_k productions will fire whenever they are satisfied. A given characterization (Data D_1 ... D_m) will last as long as no other R_k, is sufficiently similar to fire. Thus the variable duration will simply happen. The time course will be governed by the set of R_k's that are available, ie, how dense a net they throw over the space of acoustic-feature vectors. With this scheme the two varieties of variable duration merge into one. Even though the D_i move around somewhat, the refractory conflict resolution should keep the same production from firing repeatedly.

Potential difficulties certainly exist with this scheme. There is a strong dependency on the nature of the recognition match metric. Without some investigation of this metric it is unclear whether the right partitioning of the acoustic space is possible. This is simply another way of stating the concern of section (I9) that the recognition match be able to provide an adequate acoustic metric. However, this scheme seems basically sound enough as an initial pass.

(I13) Initiation of recognition. In Harpy, speech processing is initiated by a small communication protocol: the speaker initiates with a written command (Listen!); the system returns a visually displayed command (Ready!); after which it starts to receive, taking the first significant sound intensity to be the start of the utterance. Analogous to doing the backtrace only when the whole utterance is complete, this arrangement reflects the specialized experimental environment. However, we can capture the essence of the arrangement by defining a collection of *initiation* productions:

(4.23) V_k: (Data A_k) --> (State S_j S_{start} L_k)

The V_k go directly from sound patterns to putting the initial state element into WM. These latter then make contact with the appropriate $P_{i,j}$. There may be any number of the productions; in fact, compared to the rather small number of transitions from a state symbol well into an utterance, the context for initiation may be quite broad.

Missing from the V_k is any notion of the initiation depending on some external state, ie, some expectation. An alternative formulation to (4.23) would enter a single state element (State S_{start} ...), which would then trigger the equivalent of (4.23) as regular $P_{i,j}$ productions. Entering the single initial state element would easily be made dependent on expectations. However, there is little point here in expanding the system in this way.

(I14) Precision of likelihoods. Interestingly, nothing has yet forced the question of the precision of the likelihoods and the comparison. Yet this must ultimately be a major item in the plausibility of the model. Harpy currently uses an 8 bit encoding of the likelihood (ie, 256 distinct values). Experiments with Harpy showed that accuracy decreases significantly if only 6 bits are used.

So far the restraints placed on our representation of likelihood have no interaction with a requirement that likelihoods be precise to at least 8 bits. Thus we need to be much more specific about how we will represent likelihoods before we can even express this demand in a useful way.

(I15) Summary: Encoding likelihoods. We have now taken care of most of the major mapping issues. In the course of this we have accumulated a number of explicit promissory notes. Let us list them under the issues where they achieved their final form:

(I4) Avoid confusions on backpointers.

(I5) Maximize by means of recency or special case order.

(I6) Match on closeness of likelihoods and acoustic parameters.

(I7.1) The probability of match might depend on the closeness.

(I7.2) Replacement or creation of elements in evaluation.

(I7.3) Synchronization problems between generation ($P_{i,j}$) and evaluation ($Q_{i,j}$).

(I8) Increment an element according to the residual.

(I9) Summing residuals produces desired acoustic match metric.

(I11.1) Controllable lockout or no intrusion of extra backtrace starters.

(I11.2) Limited lockout on stepping through the backtrace.

(I13) Likelihoods must be precise to 8 bits.

Much of this list concerns how one passes from signals, ie, continuous quantities, to symbols, ie, expressions over discrete alphabets. How does one encode the likelihoods to have the correct properties? It was important to get all the aspects of this basic theme together, rather than to deal with each piecemeal. A moments consideration shows that the problem is much larger than the perception of speech. It concerns the general transformation of all sensory signals into symbolic form. Thus, it will pay us to turn aside and look at the general encoding of stimulus intensity. Though we can only afford to do it briefly, it will at least provide some basis for a design choice that may work out over a wider domain than just speech. With the larger perspective we will then be able to complete the mapping that we have constructed in this section.

5. THE REPRESENTATION OF INTENSITY

Continuous magnitudes show up in human cognition in several places and almost surely in different representations. We know something of the sensory transduction into neural pulse rates, which are an analog representation of stimulus intensity. We also know that humans use a wide variety of symbolic notations for quantities, eg, familiar radix notation, as well as more imperfect collections of discrete quantity names ("big", "little"). That there is signal to symbol transduction is beyond doubt; there may be several such transductions. It is a feature of all representations of the same underlying domain (here the continuum), that they are capable of mimicking each other. To first order this is built into their being representations of the same thing. Thus, we should not expect it to be easy to determine empirically what representations are being used when we have only, or mostly, behavioral evidence available.

5.1. Types of Representations

Within a production system architecture there are several possibilities for representing quantities. [7] For short we can call these the *counting*, *population*, *radix*, *value*, *dual-value* and *activation* representations.

A counting representation makes use of the ability of an expression to hold various numbers of tokens. Thus given a primitive symbol X, the expressions (X), (X X), (X X X) encode 1, 2 and 3 respectively. Given that new quantitative symbols can be defined, then (X3 X4) is 7, where X3 is (X X X), etc. This is an analog representation for the integers, making use of the physical properties of symbolic expressions.

A population representation uses the occurrence of N data elements in WM as a representation of the number N. Thus, if WM held (X A) (X B) (X C) this would be taken as meaning three X's. The representation is similar to a counting representation, but without any easy way to localize, store or manipulate the representations. It would seem hardly worth mentioning, except that it showed up in issue (I7) as a possible representation of likelihood to deal with gradual extinction of the pure generation productions (4.13).

A radix representation is a symbolic representation that has a collection of primitive quantitative symbols (the basis), such that various expressions involving the primitive symbols in known roles can be interpreted as quantities. The common number notation provides a standard example. The digits 0, 1, .. 9 are the basis and an expression (9 4 2 3 6) is interpreted as the number $9*10^4 + 4*10^3 + 2*10^2 + 3*10 + 6$. There are similar representations that have somewhat different properties. For instance, let a quantity be represented by (S_1 S_2 .. S_n) where S_1 is the best estimate that can be given with one

7 An extended treatment of the development of quantitative symbols within a production system framework has been produced by Klahr and Wallace (1976). However, it focuses on issues that do not help us with the encoding of initial intensity information.

primitive symbol, S_2 is a correction in either direction to yield the best estimate in two symbols, S_3 is another correction, again in either direction, and so on. This is radix like, but may not have uniqueness of representation, ie, $(S_1 + S_2)$ and $(S_2 - S_3)$ may represent the same quantity.

A value representation admits symbols that can be treated as if they were continuous quantities. Expressions such as (S 4.32) or (T -3) are admissible, along with (S X) or (T TEMP). This begs the question of how the quantitative symbols represent quantities. When one uses such notations in programming languages (eg, Lisp) there is another underlying representation (the bit array). But the issue can be finessed and it can simply be assumed that a quantitative value can be used at any place that a symbol can occur. These values can be treated as if there were some analogue representation with only finite accuracy.

A dual-value representation associates with every symbol a quantitative value. Each position in an expression has both a symbolic name (S1) and a value (4.32), eg, (S1:4.32 S4:-3). This contrasts with a pure value representation where numbers are simply one alternative. It agrees with the value representation in simply assuming some encoding of continuous magnitude for the quantitative symbol.

An activation representation associates with elements in WM a quantity, namely their activation, which is a continuous magnitude. This is not available in LTM (in fact, positive activation defines existence in WM). Thus it shares with the dual-value representation that elements have both symbols and quantities, but contrasts with it by the quantity only being available in the short term.

All six of these representations come in various flavors and with various restrictions. One major dimension is the representation of negative numbers and of fractions. For instance, in its natural form activation only represents positive numbers. A second dimension is the set of operations involved. Counting and radix representations do not add any new operations at all, but pose the problem of how to achieve the use of quantitative symbols through the existing capabilities of the architecture. Value and dual-value representations both require adding primitive matching predicates (or metrics) and primitive actions. Activation, as it currently exists, works through conflict resolution rather than the recognition match; it is thus quite different in its operational character from the others. However, this might be enriched in various ways to make the activation capable of being sensed and manipulated. This might even lead to providing some capability for long term storage.

In this paper there is only space to explore a single alternative for how to represent magnitude. The most appealing is the activation representation. The range of alternatives have been enumerated to indicate that other paths are possible as well.

The two purely symbolic representations (counting and radix) suffer operationally from the requirement that most of the speech computation must be done by variable-free productions. How can such productions accomplish symbolic arithmetic? At a more general level, it seems odd to have a biological system that moves from a purely symbolic machine

toward quantitative symbols as composed expressions. Surely the evolutionary movement has been in the other direction. No such objections exist for the value and dual-value representations. What raises concern here is that they will provide too much power. If, following our "natural" programmer instincts, we simply provide the full arithmetic set of operations (add, subtract, multiply and divide), then how will we explain the difficulty the human has in doing arithmetic? Even restriction to addition and subtraction still provides substantial power. One recommendation for activation is that it is already part of the architecture, being the one way in which continuum notions are already available. Also, it seems somewhat easier to provide limited capabilities, that do not blossom into awesome computational powers. None of these arguments are conclusive, for there are many variations possible of each representational type. Thus, the selection of an activation representation is a designer's choice.

5.2 Activation Encoding of Intensity We can formulate this design decision as another step in the HPSA77 specification:

> D5.1. Sensory information is encoded into the activation of WM elements.

HPSA77 already assumes that there is an amount of activation associated with each element in Working Memory. The actions, whether those on the right side of productions or those external to the production system, endow an element with an initial activation level which undergoes some time course of decay, until the element is no longer capable of aiding in evoking a production. Conflict resolution according to recency is in terms of activation. The way that activation of all the WM elements that support a production combine to yield its *effective* recency is a major degree of freedom available to make activation an appropriate way of encoding sensory intensity. Currently in OPS2 the combination rule is lexicographic order, but this is known to be unsatisfactory.

A useful way of determining whether activation will behave correctly to encode intensity is to explore how it handles the phenomena of simple stimulus detection and estimation, ie, sensory psychophysics. This will help avoid making too many inappropriate assumptions in mapping Harpy. This can be done expeditiously by considering some models that appear moderately successful in handling psychophysical data.

Two of these are the so called *Timing* and *Counting models* (Green & Luce, 1975; Luce & Green, 1972; McGill, 1967). These both assume that intensity is encoded as a series of pulses on neural channels with the pulse rate being monotonically related to the intensity. The basic measurement act is accumulating numbers of pulses and the times they take, over one or many channels. The basic decision procedures involve fixing times or counts, taking the measurements and thresholding them against criteria. Sometimes, as in magnitude estimation, the measurements themselves must be converted into a response, hence some kind of genuine transduction must occur. The two models differ in what they hold constant (times or counts) and correspondingly what they measure (counts or times, respectively). Both, however, share the property that they inherently couple accuracy and time, since more

accuracy can be achieved by taking a longer time to accumulate evidence. Thus, they directly create the opportunity for time-accuracy tradeoffs. This arises from the basic choice of representation in which intensity is coded by means of pulse rate (equivalently, pulse interval).

As analyzed in the literature, the models in general assume away the issues of central concern here. That is, they assume a homunculus to carry out the various operations and calculations required. The procedures to be carried out are determinate and though they require processing and control, do not require intelligence. One is tempted to describe them as simple procedures, but that judgment is relative to the means at hand, ie, to the architecture. The work on these models has focussed on asking how well they explain the range of known psychophysical phenomena. Our focus is on whether HPSA77 plus D5 above can implement these models in an appropriate way, ie, how it can explain the homunculus. If it does, then we inherit, so to speak, their explanations (as well as their flaws).

Consider a version of the timing model: On each of J channels count a fixed number of k pulses and sum (or average) their total times. This yields an estimate of the stimulus intensity associated with that bundle of channels. In HPSA77+D5, without further assumptions, there is no way to measure times directly, ie, no master clock. What is available is the decay of activation. Thus, to measure time a signal fires a production that activates a timer symbol (Clock), ie, simply a symbol whose activation will be used to measure time.

(5.1) T1: (Signal) --> (Clock)

The detector production is attempting to sense the existence of an event (the pulse) on another channel (Z) and to obtain the time measurement from (Clock).

(5.2) T2: (Clock) (Z) --> [time measure]

A choice is available about whether the measurements will be taken with respect to the decrease in activation from the top or in absolute amount of activation. The latter looks less attractive, since the system itself would seem to be bathed in a bed of noise, so that no absolute base line is detectable. Furthermore, if we attend to the capabilities we wish to assume for Harpy, we see that the main sensing technique is the residual formed during the match. Thus, if we take (Clock) on the condition side to represent the maximum activation, the residual of the (Clock) condition and the (Clock) in WM provides a measure of the decay of WM (Clock), hence of how long it has been ticking as a clock.[8] Thus we get a revision of T2:

(5.3) T2: (Clock) (Z) --> (Time [r((Clock))])

The residual from the match is not itself a symbol. That is, it seems unlikely that a

8 Note that (Clock) in T2 is not the same element as (Clock) in WM, so they can have different activations; we could not have used Clock (ie, the same symbol) in both expressions.

signal to symbol transformation occurs within the recognition match, such that match deviations show up as quantitative symbols somehow. Rather, the residuals seem more akin to activation, something that can be used to modify the activation of other symbols.

To pin this down we need to invent a precise set of rules for sensing and manipulating activation. This is necessary to move beyond our informal use of brackets. We now have enough context simply to lay out a particular scheme at examine its properties. These add additional specifications to design decision D5.

Let us redefine brackets to mean the manipulation of activation. Thus, X[...] means that the expression in [...] has to do with the activation on X. If it occurs as part of a condition, it has to do with sensing the residual; if it occurs as part of an action element, it has to do with setting or modifying the activation of the element. We admit variables in brackets, eg, X[=y]. These bind to activation, not to the symbols themselves. Activation variables can only be used to set or modify the activation of existing symbols. Whether those symbols are newly created or not is determined independently in the usual way in HPSA77. The use of activation variables does not evoke the UV mechanism; hence activation variables may occur in either variable-free or variable-containing productions.

We will start by assuming that only the actions of setting and incrementing can be performed on the right hand side of a production. Thus X[=y] will set X to have activation =y; X[+ =y] will increment the existing activation of X by =y; and X[- =y] will decrement it. The operations of differencing (obtaining the residual) and thresholding (match or no match) are performed by the recognition match (hence occur on the condition side of a production); they do not occur as actions. We leave open the issue of independent control of thresholds in the condition elements, ie, setting an activation level for the condition elements and symbols to be used in the match. We do assume that the activation of an element, say (A B C), is composed (simple summation) from the activation of its components plus some activation proper to the expression itself. Thus, the basic production evocation also provides for the operation of accumulation. The ability to assign activation variables is important; it amounts to a strong control assumption about independent access to the components of activation. It permits some conditions to be used to gate or control the firing of a production, while other conditions sense activation. Without this, ie, with just the total accumulated residual available to modify the action elements, there would always be a mixture of effects to cope with.

The above assumptions can be summarized as follows:

> D5.2. The activation of an element is the activation of its components plus an activation proper to the element itself.
>
> D5.3. Activation can be sensed by the recognition match; anywhere a normal variable can occur, an ***activation variable*** can also occur to sense the residual of the match to the component of the template. Activation variables do not involve the Use-Variables (UV) mechanism.

D5.4. Activation can be set, incremented and decremented by actions, using either constant amounts of activation or sensed residuals (values of activation variables).

These assumptions do not appear to modify HPSA77, but only to further specify it. One of the strongest specifications is to a particular form of recency conflict resolution. The scheme strongly suggests, though it does not appear to absolutely demand, that the recency ordering be based on the total accumulated activation from all the condition elements. This is only one form of recency rule and, in fact, not one that has been explored extensively.

We can now turn back to formulating the Timing Model. We get a new version of T2:

(5.4) T2: (Clock)[=t] (Z) --> (Time X[=t]) (Decide)

The activation variable =t is set to be the residual of (Clock) in the template, set at some standard value, and (Clock) in WM; this becomes the activation level of X within the expression to hold the time. (Decide) is a control signal telling the decision production that it is time to decide. This latter production can make the decision based on what (Time X) says:

(5.5) T3: (Decide) (Time X) --> ...

T3 does not admit of a settable threshold. One way of accomplishing this is by a second condition element:

(5.6) T3: (Decide) (Threshold V) (Time X) --> ...

By manipulating the activation on V one gets evocation or not corresponding to how big the residual is on X. Another way to set thresholds, of course, would be to introduce threshold setting operations directly. But we will try to get by without this extra apparatus.

T1-T3 handle only a single pulse on a single channel. Accumulating time for multiple counts for a single channel can be done either by holding off reading (Clock) until the k-th pulse or by adding in a residual with each pulse and quitting after k pulses. In either case this requires counting the pulses on Z. There are serious limitations to this within HPSA77, which correspond directly to the limitations of humans to do lots of very fast counting. If we postulate a good mechanism for counting at one place in HPSA77, we must then show why it could not be used in other places as well. Small amounts of counting can be achieved in several ways, eg, by state transitions or by recognizing counting patterns (such as (X X X)). But large counts remain an open issue. (It is unclear whether large counts occur in the Timing model; most existing work involves less than ten pulses per channel.)

Let us simply choose one way: counting by state transitions and then reading the (Clock). We get the following modification:

(5.7) T1: (Signal) --> (Clock) (Count T_0)

(5.8) $T2_i$: (Z) (Count T_i) --> (Count T_{i+1}) $i = 0, 1, \ldots k-1$

(5.9) $T2_k$: (Clock)[=t] (Z) (Count T_k) --> (Time X[+ =t]) (Decide)

(5.10) T3: (Decide) (Threshold V) (Time X) --> ...

The T_i are simply k+1 arbitrary counting symbols, analogous to the S_i used for speech states.

The final mechanism required is to accumulate over multiple channels. This can be done either by accumulating the activation from all channels into a single place or by keeping separate accumulators and using the accumulation feature of the condition elements at decision time. Looking at the separate-accumulator version, there are J channels, Z_j, so the $T2_i$ get multiplied to run off J separate counters (but using the same (Clock)) and T1 gets multiplied to initialize the J counters. There must also be J (Decide) elements to record each channels reaching the k-th count and T3 must fire only if all channels have finished. We get the final system:

(5.11) $T1_j$: (signal) --> (Clock) (Count$_j$ T_0)

(5.12) $T2_{i,j}$: (Z) (Count$_j$ T_i) --> (Count$_j$ T_{i+1})

(5.13) $T2_{i,k}$: (Clock)[=t] (Z_j) (Count$_j$ T_k) --> (Time X_i[+ =t]) (Decide$_j$)

(5.14) T3: (Decide$_1$) ... (Decide$_J$) (Threshold V) (Time X_1) ... (Time X_J) --> ...

There are still some uncomfortable things about (5.11) - (5.14), eg, the multiple counters and the large number of condition elements in T3. But we have hardly explored the space of productions systems, and substantial tuning of the system is possible. We need not pursue the details further.

However, some remarks are appropriate about general plausibility and sufficiency. The productions are variable-free, so that the entire process can occur concurrently, which is clearly necessary. However, important demands on precision of activation are set by this scheme, and they may prove too stringent. The extensive character of the productions, with all the separate counters and counter states, may have consequences for establishing such systems, a matter that we haven't gone into at all. Though we did not present a version of the Counting model; its character follows along similar lines, given the development above. Thus, whether a Timing model or a Counting model is used is clearly a matter of software, and is not, so to speak, wired in. This is what Green and Luce (1975) found, namely, that payoffs could swing human subjects from one kind of behavior to the other. Though incidental to their focus of fitting specific models, such considerations are critical from an architectural concern.

In any event, we can rest here. This excursion was intended only to provide a crude indication that adopting decision D5 would be a reasonable choice. (5.11) - (5.14) show that

decision D5 has a chance of being commensurate with the Luce and Green Timing model for psychophysics, and thus being a plausible model of sensory intensity generally.

6. HARPY AS A PRODUCTION SYSTEM: FINAL VERSION

Accepting the design decision of the prior section on the representation of intensity, which identifies sensory magnitude with activation level, we can rework PS.Harpy.

6.1. The Final Version.

The main issue is what activation to identify with the likelihood in the state expressions, (State S_i S_j L). There are three immediate possibilities: the special symbol L; the state, Si; or the state expression (State S_i S_j L) itself. As usual, the right choice is not completely clear. However, L seems redundant with the state expression, and the use of the state symbol would appear to produce a classic type-token problem if the same state were to be considered at more than one time in the utterance. Use of the expression itself has the virtue of simplifying the state element by eliminating L, since it is redundant. To be explicit, we add another mapping assumption:

(M6) Likelihoods are encoded as the activation on the state element.

With this stipulation, we can simply write down the new production system for PS.Harpy:

(6.1) $P_{i,j}$: (State S_i =)[=l] --> (State S_j Si)[=l - $k_{i,j}$]

(6.2) Q_i: (State S_i =) (Data A_{i1} .. A_{im})[=d] --> =(State S_i =)[+ =d]

(6.3) R_k: D_1 D_2 .. D_m --> (Data D_1 D_2 .. D_m)

(6.4) U1: (State =S =) (Data A_s) --> (Utterance S_{end} =S)

(6.5) U2: (Utterance = =S) (State =S" =S') --> (Test =S =S' =S")

(6.6) U3: (Test =S =S' =S') --> (Utterance =S =S')

(6.7) W_i: (Utterance S_i =) --> (Word Wx)

(6.8) V_k: (Data A_k) --> (State S_j S_{start})[l_k]

(6.1) arises from (4.13). It generates the next state elements without regard for the acoustic data. Representing likelihoods by activation now permits a decrement on pure extrapolation that retains the memory of the likelihood of the input state. The decrement itself can be a function of the transition (S_i to S_j), as expressed in $k_{i,j}$. Thus, the worries about history-free generation that attended our initial formulation have vanished. Depending on the size of the decrement ($k_{i,j}$), acoustically unsupported extrapolation will eventually come to a halt.

(6.2) arises from (4.15). It operates to select out those forward paths that are supported by the acoustic data. It does this by boosting the likelihood by the amount of the acoustic match (+ =d). This increment is added to the existing likelihood element and not to a newly created element. All (6.2) does is enhance good paths. The actual clipping of the beam occurs because state elements with too low an activation level will not evoke their $P_{i,j}$ productions to get extended.

Note that (6.2) has Q_i, not $Q_{i,j}$ as in (4.15). The productions are the same; the question is whether one can get by with a single production for each state (S_i) or must have one for each transition, ie, each second state symbol (S_j) in the production. The underlying question is whether a variable-free production will be called on to execute on multiple instantiations. But if Q_i is satisfied by many state elements (with the same S_i, but different S_j), then recency conflict resolution will let it fire on only one, the maximum. Hence, a single Q-production can be used per state.

Similarly, it can be seen that the argument in section (I5) goes through that the maximum will be taken for sets of transitions to identical states, as required by equation (3.3). Two state elements leading to the same state will occur as (State S_k S_i)[=l_i] and (State S_k S_j)[=l_j] with =l_i < =l_j. $P_{k,m}$, which extends the path to (State S_k S_m), will fire in any event. Recency (ie, maximum activation) will assure that the activation level of the match will be that of =l_j, the maximum input likelihood.

(6.3) arises from (4.22) without change. It operates to segment and chunk the context-free representation of the auditory input. If the features D_i are sufficiently activated to evoke an R_k, then an acoustic description of the segment (Data D_1 ... D_m) occurs in WM with high activation, being the most recent one.

(6.4) - (6.7) arise from (4.17) - (4.21) without change, except to delete the likelihood symbol. They perform the backtrace and create a sequence of words-in-context in WM. As discussed earlier, this sequence is the putative output of the process of speech recognition, to be used in various cognitive ways.

(6.8) arises from (4.23) without change, except to accommodate the activation encoding of likelihood. It initiates speech processing.

The system of productions (6.1) - (6.8) constitutes the final form of the mapping of Harpy into HPSA77. It is not a complete system of productions, for it does not perform all the tasks needed for Harpy. It takes the isolated acoustic parameters as given input, but these must come from somewhere. It produces a conventional output from the backtrace rather than one appropriate to some actual task to be performed. It provides only for the performance system of perceiving speech, not the system outside the net that contains grammatical, lexical and phonological knowledge or the system that compiles this prior knowledge into the state network. Though a small point, no additional control structure has been provided to let speech perception occur in the midst of the general cognitive life of the perceiver. For instance, WM is filled with other symbols concurrently with recognizing the

utterance. Still, this production system, PS.Harpy, is complete enough to permit some assessment of the enterprise.

6.2. *Analysis of Basic Sufficiency Questions.*

With (6.1)-(6.8) in hand, we consider the basic sufficiency questions posed in Section 3.2.

PS.Harpy is an attempt to map faithfully the current Harpy into HPSA77. By so limiting our aim, we obtained a well-defined task, which did not mix questions of interpreting the mechanisms of Harpy either with new ideas for speech processing or with modifications to explain known human speech phenomena. We also preserved maximally the sufficiency of the system to perceive speech (to the limit of Harpy). Redesign and embellishment begins to loose Harpy's demonstration of adequacy and to evolve toward being simply another (new) proposal for a speech system.

However, as we now confront the various ways in which PS.Harpy is inadequate, either on sufficiency grounds (in this section) or explanatory grounds (in the next section), we should consider how PS.Harpy could be modified and extended. There is nothing sacred about an exact mapping of Harpy, especially about the particular sequence of mapping decisions used. The chief virtue of (6.1)-(6.8) is to provide an initial point in an approximating sequence of theories of human speech perception, a point that has certain desirable properties, namely sufficiency in the sense we are using the term.

(S1) Can PS.Harpy recognize speech? PS.Harpy is as adequate as Harpy, providing that the mapping is completed in a way that achieves the remaining details of Harpy. In section (I15) we listed a number of promissory notes and open choices. Making decision D5 and mapping assumption M6 has resolved many of these issues. But the remaining ones constitute the requirements for appropriate further specification. They are the still open questions about the performance side of sufficiency.

(I4) Avoid confusions on backpointers.

(I7.3) Synchronization problems between generation ($P_{i,j}$) and evaluation (Q_i).

(I9) Summing residuals produces desired acoustic match metric.

(I11.1) Controllable lockout or no intrusion of extra backtrace starters.

(I11.2) Limited lockout on stepping through the backtrace.

(I13) Likelihoods must be precise to 8 bits.

We will simply accept these six items as continued promissory notes, awaiting much greater detail on how a PS.Harpy would be implemented. There is no doubt that some of them (eg, the accuracy one) could threaten the entire edifice. Most of the others are subject to being solved by design improvements, eg, the synchronization and lockout problems.

(S2) Can PS.Harpy be extended to full speech? From the analysis of Figure 8, we can compute the number of productions that are required. The $P_{i,j}$ contributes one per transition (SB). The Q_i contributes one per state (S). The R_k constitute the phonetic alphabet. The 98 phone templates of the current Harpy is not a good guide, since one way of compensating for a weak metric is to expand the alphabet of templates. The International Phonetic Alphabet (with perhaps several hundred symbols when expanded) might be a more suitable guide. Thus, there might be at most a few thousand of these. This would provide of the order of ten to twenty variations per phoneme, which seems quite sufficient. The backtrace requires one production for each word-in-context (ie, Sg) plus three more (U1-U3). Finally, the initialization productions (V_k) can be viewed as providing multiple entries to a subset of the grammar nodes. We have no way of estimating this number; perhaps taking it of the order of the vocabulary (V) would not be too far off. Thus we get, in toto:

(6.9) Number of productions = SB + S + [Phonetic-alphabet] + (Sg + 3) + V

The actual numbers are shown in Figure 9 for the same range of effective vocabulary (Bg) considered earlier. (V is not uniquely determined by Bg, so we assumed reasonable values of redundancy around .6.) At the low end we get less than 10^5 productions, which might be tolerable. At the high end we get 10^{14} productions, which seems utterly out of the question. It is clear we need to consider what might modify these numbers.

The simplest extrapolation of Harpy takes the basic structure as fixed. The language itself, as characterized by V (vocabulary), r (redundancy), and the word parameters (Bw, Dw, Sw), is also fixed. Thus, the potential candidates for modification are Dg (depth before backtrace), c (grammar compression) and f (beam fraction). Considering (6.9), the controllable item is Sg (since S = SwSg), which depends on c and Dg. We know very little about c; those working on Harpy think it might decrease some with increasing language (Reddy, personal communication). We might examine a drop by a modest fraction (15%). Dg, on the other hand, has been set in Harpy to the full length of the utterance (ie, 2 sec on average) without much analysis. This might well be substantially shorter. We might examine cutting Dg to 1 second, ie, 3 words. Figure 10 shows the revised figures for Harpy. Looking at the high end (Bg = 320), we see that Sg has come down to 10^6, making S less than 10^7, so that, as shown in Figure 11, we get $5*10^7$ for the number of productions. The other numbers are correspondingly lower, as the size of the language decreases. These are still large numbers, but they are conceivable.

(S3) Can PS.Harpy recognize full speech in time? We need to compute two quantities, the total amount of computation and the total elapsed time given the concurrency available. As to the amount of computation, we need to fill in the content of the unit computation on each Harpy cycle. A $P_{i,j}$ will be executed for every transition from the existing beam, getting

Grammar Branching (Bg)	10	20	40	80	160	320
Productions	7*10↑4	5*10↑6	3*10↑8	2*10↑10	4*10↑12	1*10↑14
Firings (Harpy cycle)	6*10↑2	4*10↑4	3*10↑6	2*10↑8	2*10↑10	1*10↑12
Firings (sec of speech)	9*10↑3	6*10↑5	7*10↑7	3*10↑9	2*10↑11	2*10↑13
WM load	4*10↑3	3*10↑5	2*10↑7	1*10↑9	7*10↑10	5*10↑12
Fixed parameters:	Bw = 2	Dw = 5	Sw = 8	Dg = 6	c = .35	f = .01

Figure 9. Vital statistics for PS.Harpy

Grammar Branching (Bg)	10	20	40	80	160	320
Grammar States (Sg)	3*10↑1	2*10↑2	2*10↑3	1*10↑4	1*10↑5	9*10↑5
Harpy States (S)	2*10↑2	2*10↑3	1*10↑4	1*10↑5	9*10↑5	7*10↑6
Branching factor (B)	3.2	3.6	4.2	4.8	5.5	6.3
Computation units	1*10↑1	9*10↑1	9*10↑2	8*10↑3	7*10↑4	7*10↑5
Beam (F)	2*10↑-1	2*10↑0	1*10↑1	1*10↑2	9*10↑2	7*10↑3
Integrated Beam (Fx)	3*10↑0	3*10↑1	2*10↑2	2*10↑3	1*10↑4	1*10↑5
Fixed parameters:	Bw = 2	Dw = 5	Sw = 8	Dg = 3	c = .30	f = .001

Figure 10. Estimates of critical parameters for advanced Harpy

Grammar Branching (Bg)	10	20	40	80	160	320
Productions	5*10↑3	2*10↑4	9*10↑4	7*10↑5	6*10↑6	5*10↑7
Firings (Harpy cycle)	4*10↑0	1*10↑1	7*10↑1	6*10↑2	6*10↑3	5*10↑4
Firing (sec of speech)	6*10↑1	2*10↑2	1*10↑3	1*10↑4	9*10↑4	8*10↑5
WM load	3*10↑0	3*10↑1	2*10↑2	2*10↑3	1*10↑4	1*10↑5
Fixed parameters:	Bw = 2	Dw = 5	Sw = 8	Dg = 3	c = .30	f = .001

Figure 11. Vital statistics for PS.Harpy based on advanced Harpy

FB firings. However, roughly speaking, the Q_i will be executed only on those state elements that are sufficiently close to become part of the new beam, getting F. (This is only a rough estimate: the beam also contains elements that have not been boosted by a Q_i production (pure extrapolations); and not all boosted elements will be included in the beam.) In addition, one R_k fires in recognizing the acoustic wave as a speech sound. In the backtrace, there are two executions, U2 and U3, and possibly another one, a W_i (call it three in total). These occur much later, but can be associated with a unique Harpy cycle. The initiation productions, V_k, contribute nothing per cycle, constituting in essence a distinct cycle. Thus we get:

(6.10) Production firings = FB + F + 1 + 3 per Harpy cycle

The actual numbers are shown in Figure 9. Again, at the high end the numbers come out very large, ie, 10^{12} firings per Harpy cycle. These can be multiplied by about 15 Harpy cycles per second to get the computation per second of speech, as shown.

If we consider how this can be cut down, the same assumptions as used above reduce S. In addition, the beam fraction f is also controllable (though it does not affect the number of productions). This is directly responsive to how good the acoustics are -- the more discriminative and reliable, the smaller f can be. The best indication of the human potentiality for pure speech sounds comes from the performance of Victor Zue (Cole, Rudnicky, Reddy & Zue, This proceedings). Reddy estimates that this might result in a factor of 10 improvement in f, which is to say f = .001 instead of the current .01 for Harpy. Extrapolated estimates are shown in Figure 10 for Harpy and in Figure 11 for PS.Harpy. For the maximum case we now have about $5*10^4$ firings per Harpy cycle. Again, these figures are high, but not out of the question.

Given the complete concurrency of the PS.Harpy scheme, it is not clear that an important constraint can be derived from the number of production firings. PS.Harpy is posited to be capable of firing any number of variable-free productions concurrently.

The second computation we need is the amount of elapsed time to recognize speech. The key issue is what production firings must occur sequentially, either because the output of one is the input to another or because they use the limited resource of the UV mechanism. Figure 12 lays out the pattern of production firings. The development of the acoustic signal by the R_k paces the computation, but occurs concurrently. Only the first occurrence, if any, contributes to the elapsed duration. A single firing of a V_k is required to initiate the forward sweep. The forward sweep itself consists of pairs of associated $P_{i,j}$ and Q_i productions. These occur in series: P generates, then Q boosts, to provide for the next P to generate. The forward sweep consists of D Harpy cycles, so there will be D firings of P productions and D firings of Q productions. The backtrace must follow this same path, after a single firing of U1 to initiate it. U2 and U3 pairs execute in series for D steps. Both U2 and U3 are variable-containing productions. However, their sequential nature arises from the data dependence as well as from the limited resource of the UV mechanism. The W_i, which bring down the word codes, are variable-free productions and operate concurrently with the main backtrace. Presumably the last one occurs after the last cycle of the backtrace and could add

to the elapsed duration. Thus, using T[X] to stand for the duration of a production of type X, we get:

(6.11) $$\text{Duration} = T[R] + T[V] + D*T[P] + D*T[Q] + T[U1] + D*T[U2] + D*T[U3] + T[W]$$

How long these production firings take depends on the detailed timing assumptions about the recognition-act cycle: the recognition time, the action time, the timing effects when variables are involved (ie, UV is evoked). These assumptions have not been made yet in HPSA77, though the UV mechanism was assumed to add around 40 ms per instantiated action (independent of recognition time) and recognition time was assumed probably to be longer. In particular, nothing was specified about possible processing time differences between variable-free and variable-containing productions. Besides the uncertainty on component time assumptions there is also the question of how many instantiations will be created by U2 before the lockout terminates and lets U3 fire.

Even with these uncertainties, we can express (6.11) in terms of basic recognition and action times. We subscript the times according to whether they are variable free (vf) or variable containing (vc); and let L be the number of instantiations that occur within the lockout for U2.

(6.12) $$\text{Duration} = (2D+3)T[\text{Recog}_{vf}] + (2D+3)T[\text{Action}_{vf}] + (2D+1)T[\text{Recog}_{vc}] + (LD+D+1)T[\text{Action}_{vc}]$$

Some insight into (6.12) can be obtained by transforming it to express the duration required per second of speech. Humans keep up with speech input (and in fact with speeded speech); which implies that they must process speech in less than real time on the average. (6.12) is the amount of time to process D segments. However, it cannot just be normalized, since if continuous speech is being processed the forward sweep of one recognition phrase can be processed concurrently with the backtrace of the prior one. This is shown on Figure 12 by the bottom line of Ps and Qs occurring under the Us in the main computation. What counts then is which of the two sequences takes longer, the P-Q sequence or the U sequence. Assuming it is always one or the other (just for simplicity here) we get:

(6.13) $$\text{Processing time per segment} = \text{Max}(2T[\text{Recog}_{vf}] + 2T[\text{Action}_{vf}], 2T[\text{Recog}_{vc}] + (L+1)T[\text{Action}_{vc}])$$

We have suppressed terms that are proportional to 1/D; they are small and have no effect on which component is the maximum.

It seems extremely unlikely that the times for recognition or action of variable-free productions will be less than those that use the UV mechanism. Consequently, it can be seen that the backtrace will always dominate (6.13). Thus, we can simplify:

(6.14) $$\text{Processing time per segment} = 2T[\text{Recog}_{vc}] + (L+1)T[\text{Action}_{vc}])$$

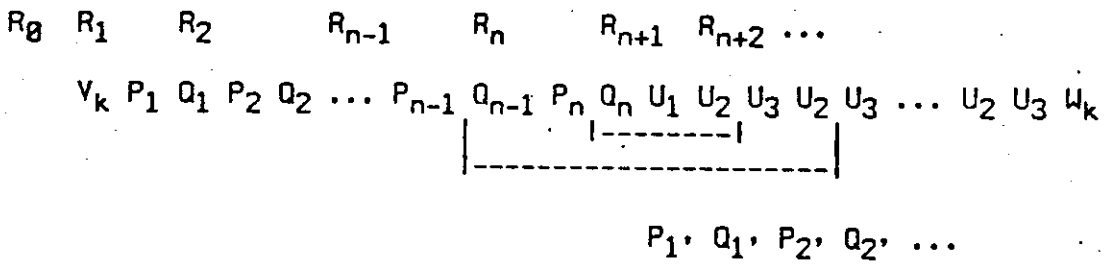


Figure 12. Elapsed time to perceive an utterance

Segments in Harpy run about 50 - 70 ms (depending on whether one counts the input segments or their lengthening by the algorithms that perseverates in a state). Taking recognition time as 80 ms and action-time as 40 ms, just for a sample calculation, we get (200 + 40L) ms for 50-70 ms of speech, or a ratio of over 3 times real time at a minimum. This says a second of speech takes at least 3 seconds to recognize. This is too high by a factor of 3. If we check the minimum component (the forward sweep) we see that it consists of 2 recognition times and 2 actions times per segment, but of variable-free productions. Turning the question around and asking what primitive times would be acceptable, we get about 25 - 35 ms per total production (a recognition plus a single action) for variable-free productions. This might be quite acceptable, given that the UV mechanism is not involved. In any event, given the uncertainties in the specifications, we should not be unduly concerned with discrepancies of the order we have here. From a sufficiency point of view, these are quite satisfactory. However, this does post notice that a strong constraint on the theory is available from a consideration of speech rates.

As important as the quantitative measures are the qualitative questions of what might affect the duration of the computation. The duration is strongly dependent on the segmentation of the speech signal by the R_k productions. If they produce more or less segments per second, the recognition time will respond proportionately. Beyond this we need to distinguish clearly questions of total duration, as seen in (6.12), and questions of processing rates, as seen in (6.13). In one the forward sweep and backtrace add to determine a total time; anything that effects either component induces an effect to the total time. In the other only the maximum component counts; anything that effects the lesser component has no effect on the total at all.

Looking at the forward sweep, it is possible that not all the Q_i fire. They will not fire during gaps and they may not fire in other cases as well, eg, high likelihood paths may race ahead without needing to wait until the Q-boost occurs. This would speed recognition durations; it would seem not to effect processing rates since the forward sweep is the minority component.

Whether the backtrace could be speeded up is a complex matter, though clearly important since it appears to be the limiting process. The obvious tactic is to build backtrace productions that pick up several chained state elements at a time. This is precluded by D1.4, which will not let a variable occur in more than one condition element. This is a good example of the force of D1.4. If there were no such limitation on the power of the match, then productions with, say, 15 chained condition elements, could pick up the entire speech path in a single cycle. However, one can pick up several such chains as free paths and encode them all in a test element, just as U2-U3 operate. Then the correct test element could be selected out in one recognition cycle. It takes time to do intermediate chunking, so detailed investigation is needed to discover what the real possibilities are. Beyond this, there may be strategies for heuristically guessing the path, possibly by trading accuracy against time. For instance, one could jump directly from the end of a word (as picked up by the backtrace) to the front of the word, avoiding all the intermediate states. At some point in this reduction the backtrace may drop below the forward sweep, and thus no longer be the critical determiner of processing rate.

There seems little point in exploring this space of algorithms in detail here. We have established that processing rates do not remove PS.Harpy from consideration. Indeed, they make PS.Harpy seem quite plausible, in comparison with the size estimates. This is not surprising, moreover, since the use of concurrent processing trades off memory for time.

(S4) Does PS.Harpy require too much immediate memory? Two memory loads need to be considered: the maximum amount of information stored at any point; and the total duration over which storage must occur. As to the total load, essentially this will just be the extended beam, ie, one data element for each state in the beam over long enough to initiate the backtrace, ie, FD elements. It might be thought that more must be remembered, since the backtrace itself takes time and the speech keeps coming in. But the backtrace runs concurrently with the (new) forward wave, so this requirement does not occur.

As to the total duration of any element, this must be the interval to initiate the backtrace, as above, plus the time it takes the backtrace to get though the whole history to the initial element. Since the utterance is fully recognized the moment the backtrace is complete, this is just the total processing time for the D segments, ie, (6.11). As in the discussion above of processing time, until we have a closer estimate of the recognition and action times we cannot draw strong quantitative conclusions. Above, we did get an minimum estimate of 3 times real time for the backtrace with something less than this for the forward sweep. That might add up to anywhere from 4-6 times real time for the total duration. This would yield 4-6 seconds of immediate memory, since we have been assuming D to be about a second of speech. This is again too large a value, and in about the same way as the processing rate. But it can certainly be taken as within the range of plausibility, given the uncertainty in the calculations.

(S5) How does PS.Harpy cope with the creativity and variability of speech? In the original discussion of (S5) we posed the issue of creativity and variability in speech perception by enumerating several concrete task that required transcending the fixed network in some way. Two basic capabilities seem to be involved: (1) how to make use of knowledge about speech that is already in the fixed network, but embedded in a way that seems to preclude its use; and (2) how to make use of added knowledge that has not yet been integrated into the network. This latter knowledge could already exist in long term memory (ie, assimilated as productions) or it could just have been acquired and be only in WM (ie, not yet exist as productions). To first approximation we can ignore this difference, since any way of affecting the perception subsystem will work through data elements placed in WM.

Consider the first capability: making use of the knowledge already in the net. Such knowledge is encoded as subnets (or sets of subnets generated according to systematic rules). The knowledge can be hidden in the net because access to the knowledge is not available, eg, there is no transition in the net from existing states in WM to the given subnet. Knowledge might also be effectively hidden if gaining access to it, though possible, leads to undesirable effects, eg, taking outgoing transitions to inappropriate states.

On the access side, the V_K productions already provide a basic solution. They give access to an arbitrary Harpy-net state from an acoustic clue. Thus, given the appropriate V_k, PS.Harpy can make use of any subnet; it is not restricted to entering through the transitions already available in the net. The net effect of such V_K evocations is to add to the size of the beam. But such additions would come on at relatively low likelihoods and get eliminated quickly unless subsequent states pickup strongly or (importantly) unless all the existing next states are failing to gain any support, so that the new path rises rapidly towards the top of a relatively weak beam.

At first glance it would seem that, once in the net, no bad side effects could come from continuing as long as possible. Thus, knowledge could not become unavailable because it leads to following inappropriate paths. After all, PS.Harpy wishes to understand whatever is linguistic utterance is being heard. Perhaps the V_K are all the mechanisms that PS.Harpy needs to make use of all the knowledge in the net.

Two counters to this are immediately apparent. First, nothing has been said about the mechanisms for acquiring the V_K. This is a valid concern. What little we have to say about it is best taken up under issue (S7) below. Second, knowledge may be available that sharply restricts the speech (eg, only three words were spoken). Following the natural transitions in the network violates such knowledge; hence, a priori useful pieces of the network become unavailable. Again, this is a valid concern. Such restricting knowledge is necessarily of a transitory nature and the problem of using it belongs to the second major concern we expressed above, to which we now turn.

Consider, then, the second capability: Making use of knowledge outside the network. Such knowledge could be the original linguistic knowledge from which the network was compiled; or it could be extra linguistic task knowledge (as in our example above). A basis for this capability lies in the ability of productions to execute indirectly. Consider first the knowledge about transitions, which is contained in the $P_{i,j}$ productions. We encode it in an element, (Indirect (S_j S_i)), which other productions can place in WM. Then the following two productions perform the same function as the $P_{i,j}$:

(6.15) X1: (State =S =) (Indirect (=S' =S")) --> (Test' =S =S' =S")

(6.16) X2: (Test' =S =S' =S) --> (State =S' =S)[1]

If a state element (State S_i S_h) shows up in WM and a corresponding (Indirect (S_j S_i)) element is also in WM, they will inject into WM the result of a state transition: (State S_j S_i). From this point on it will be indistinguishable from any other elements in the beam, ie, the Q productions will boost it, other P's will add further transitions, etc.

The one way in which X1 and X2 do not mimic the $P_{i,j}$ is in the assignment of likelihood. With our design choice of encoding likelihoods as activation, there is no way to symbolize this in the encoded element and assign a specific activation to the new state element, as the P-productions do. X2 assigns a likelihood, but it does so independently of which particular state transition it is making.

This issue of independence also infects the situation with respect to the knowledge in the Q_i. The whole purpose of the Qs is to boost the activation level. We can have productions X_k which provide such boosts:

(6.17) X_k: (State =S =) (Evaluation ...)[=e] --> =(State =S =)[+ =e + I_k]

However, the effects of these productions do not stem from their acting as indirect interpreters of the external knowledge (the evaluation elements), but only in the *activation* they have associated either with these elements (the =e) or with themselves (the I_k).

Putting to one side the complexities that arise from these limitations, the X-productions provide a way of injecting external knowledge that has been generated (to WM) by some cognitive process (eg, operating on "Only three words will be spoken") into the ongoing process of recognizing speech. These cognitive processes (themselves production systems) are outside our consideration here except for how they interface with the speech system. The difficulty that remains is that the Xs imply knowledge of the actual states of the Harpy network. These states would seem to be "internal" symbols of some sort and a legitimate concern is how cognitive productions gain this knowledge so they can create the appropriate WM elements.

Solutions to this latter problem would propel us into a full scale design effort for the system that compiles productions. Such an extension exceeds the bounds of this paper. However, one direction for such a design is to unpackage the pure state symbols, S_i, and replace them by descriptive elements that preserve much of the source knowledge. For example, instead of S_i one might have (Node H G_i F_i E_i), where the H is a symbol that describes the role of the state (eg, beginning, interior, or ending state for a word), G_i is a symbol from the grammar net, F_i is a symbol from the phone net for word, and E_i is a symbol for the unique conditions that cause state splitting because of word juncture, etc. Then the X-productions are modified to detect aspects of (Node ...) that they can reasonably know about (eg, a grammar node symbol).

The X-productions contain variables and this produces an important limitation. Although they can be sensitive to the entire beam, they cannot operate in lieu of the variable-free beam computation. Only a few executions of the Xs are possible within any small period of time.

These mechanisms make available some capability for accessing knowledge embedded in the network and for mixing in external knowledge. We can now dispose quickly of the various tasks posed in issue (S5) from the standpoint of basic sufficiency, ie, whether these tasks imply fundamental processing difficulties for PS.Harpy.

First, to understand grammatical utterances that have not been heard before, the problem for PS.Harpy is to extract from the net as large subpieces as are familiar. The V_k permit some of this. In addition a small amount of aid can obtained from the X-productions working on the generative knowledge of the grammar that can be presumed to exist. But no

mechanism has been suggested that permits large amounts of extra-net knowledge to be brought to bear. So whether the suggested mechanisms (along with very large nets) are sufficient must remain open.

Second, to understand an arbitrary sequence of clearly spoken words from the vocabulary is partly the question of whether the vocabulary knowledge which exists in PS.Harpy can be brought to bear. The mechanism of the V_k suggests this is possible. But recognition without grammar is clearly a task with less restraint. Performance of both PS.Harpy and of humans would be expected to decline (and it does in fact for both); nothing here permits seeing whether special difficulties for PS.Harpy, serious enough to threaten a sufficiency analysis, exist in that decline.

Third, to understand utterances that have genuinely new elements requires an involvement of the cognitive system that is beyond discussion here. The responsibility of the perception subsystem would be to obtain a sequence of (essentially ungrammatical) words and we have covered this in the second task. The ability to permit a small amount of additional analysis bases on the knowledge developed by the cognitive system during the course of inference is clearly permitted, but it is strictly limited in character.

Fourth, to incorporate immediate increments of linguistic knowledge is exactly what we have discussed for the X-productions. We have to go much beyond this analysis to see whether the mechanism is sufficient.

Fifth, to cope with grammatical errors and fragmentary utterances several ingredients are available. Such errors cause the acoustic sequence to break contact with the net. They will leave a short grammatical fragment available to be picked up by the backtrace. (The break point itself will not reflect the error sharply, due to the extrapolative character of the $P_{i,j}$, but that probably matters little.) The V_k will soon initiate another sequence, if only the recognition of isolated words. We can assume, though we haven't put it in explicitly, that the backtrace will be triggered by such initiation. Thus, the net effect is to present the cognitive system with a sequence of short fragments for interpretation. This act of interpretation lies outside the scope of PS.Harpy. Whether the preliminary recognitions that PS.Harpy does provide are sufficient for the abilities humans exhibit on ungrammatical utterances cannot be determined at this level of analysis.

(S6) Can PS.Harpy respond adequately to speaker variability? Harpy has a single mechanism for coping with speaker variability at the acoustic level, namely, computing unique phone templates and switching to a user-unique template set. Though nothing prevents adding new words or grammatical rules and constructing speaker-unique networks, these approaches have not been seriously explored.

PS.Harpy doesn't change this picture in most respects. On the positive side, the learning does not appear to be so monolithic (more on that in issue (S7)). Individualized versions of any of the productions are possible, simply by the addition of speaker-dependent condition conditions. Eg, an R_k can be of the form:

(6.18) R_k: (Speaker Z) $D_1 \ldots D_m$ --> (Data $D_1 \ldots D_m$)

The speaker identification, Z, can clearly be an individual name or a description of a class of speakers.

Adjustments on the cues that evoke a given (Data ...) are also possible:

(6.19) R_k: (Speaker Z) $D*_1 \ldots D*_m$--> (Data $D_1 \ldots D_m$)

The $D*_h$ are idiosyncratic acoustic features, which are not just chunked by the R_k, but also mapped into values, the D_h, which are already tuned to the learned structure of the net.

Similar modifications are possible for the P and Q productions. Indeed, from the viewpoint of participation in the perceptual computation, modification of any of these aspects seems essentially equivalent. Furthermore, the modifications do not have to be total, since in any event they would be represented by the aggregate of a bunch of individual production creations themselves.

On the negative side two items, at least, are serious. The first is that the only modifications possible are in productions. No factorization of the speech system into mechanism plus parameters exists such that setting a (small) number of parameters will accommodate the perceptual system to the speaker. Further, this character is built deeply into PS.Harpy (though not into Harpy itself), since parameterization implies the use of variables. Ie, for a production to make use of parameter values requires that it handle that value indirectly (as in X2 and X3). We all have, I think, strong feelings that factorization is essential for short term adaptation. Thus, the proposed scheme of adaptation seems cumbersome in the extreme in this respect.

The second difficulty is how the productions are to be learned. The mechanism above addresses the final structure for expressing speaker-dependent performance. This learning question is indeed important. We have tripped over it several times already, deferring it to issue (S7). Before turning to it, at last, we can summarize by saying that PS.Harpy leaves the sufficiency issue of speaker adaptation about in the same state it is for Harpy, but without strong indications of deep trouble.

(S7) Is language acquisition possible with PS.Harpy? Harpy's large highly detailed precompiled network seems especially vulnerable in a sufficiency analysis. Unfortunately, two issues limit our treatment. First, the context of learning involves a vastly enlarged cognitive world that must be treated concurrently with PS.Harpy. Though the question of interest is whether the compiled network violates some basic features of the human processing system, it cannot be explored very well without bringing in the entire framework within which learning takes place. Such an investigation cannot be avoided ultimately, but it can hardly be attempted as an add-on to this paper.

Second, the development of the underlying production system architecture (HPSA77)

has not yet been extended to long-term learning. This requires design decisions about the formation of new productions and their addition to Production Memory. Much current research on productions systems does involve systems that learn by adding productions (eg, Rychener & Newell, 1978; Waterman, 1975). Some of this work is directly psychological (Anderson, Kline & Beasley, 1977; Anzai, 1978; Langley, 1978; Neves, 1978). The basic learning assumption is made throughout that productions are created by the actions of other productions, ie, deliberately. This does not yet seem the right design decision for HPSA77. Consequently, the basic learning decision remains in abeyance. As in the first issue, such decisions cannot be avoided ultimately, but they are too much to add to the paper. The positing of the intensity aspect (D5), which was necessary to complete the performance model of PS.Harpy already required a detour.

Nevertheless, some aspects can be explored. The final learning of the net is clearly in terms of productions. PS.Harpy will have to construct and store away in production memory 10^5 or so productions (see Figure 11). Three questions at least are to be asked about this: (1) can that many productions be created and installed in the time available; (2) is the information available to construct the productions; and (3) are the occasions available to construct the productions? Note that other a priori obvious questions do not arise. For instance, PS.Harpy does not face a problem of how to install a production vis a vis the existing productions. Production memory is just an unstructured set of productions, so adding a production entails no complexities.

On the amount of time to learn, times for human verbal learning, which are of the order of 2-10 secs per chunk (Simon, 1974) provide for a trial calculation. The appropriate identification is that a chunk corresponds to a production (actually, such a correspondence lies buried in decision D3, though we have not made that explicit here). Taking the upper figure of 10 sec gives 10^6 sec of learning time, which is of the order of 300 hours. Though only a small fraction of a day would be devoted to such learning, this is still not an alarming number. (On the other hand, if we upped the number of productions many additional orders of ten, we clearly could be in trouble.)

The usual concern about such learning rates based on verbal learning is whether they are too slow (the process being somehow too "rote"). Shifts to faster learning pose no threat here. More of concern is the evidence that learning "automatic" responses takes thousands of trials (Neisser, Novick & Lazar, 1963; Shiffrin and Schneider, 1977). These responses should correspond to variable-free productions. However, so little is understood (or even conjectured) about what is being learned in these situations and why it takes so long, that we can do no more than note the issue.

On the other two questions of whether the information and the occasions are available, the X-productions (6.15 - 6.17) cast this issue in a specific form. If appropriate data structures, eg, (Indirect ...), (Evaluation ...), get into WM then the system can behave as if it has the productions. Thus the act of constructing a production for the net is the act of going from the X-productions plus the data elements to a production that has removed the indirect addressing capability but retains the speech-relevant parts. To illustrate with the

simplest case, given an X_k and a working memory that contains an appropriate state element for S_h and evaluation data element, the problem is to get the corresponding Q_h production. Ie, from (6.20) and (6.21), get (6.22):

(6.20) X_k: (State =S =) (Evaluation (Data D_1 ...))[=e] --> =(State =S =)[+ =e + I_k]

(6.21): WM: (... (Evaluation (Data D_1 ...)) ... (State S_h S_j) ...)

(6.22) Q_h: (State S_h =) (Data D_1 ...)[=e] --> =(State S_h =)[+ =e]

It can be seen that all the requisite structures are available to perform the construction of Q_h. Whether there is anything hard about it depends on aspects of the production-learning process that have not been specified, eg, whether details of X_k are available or are inaccessible within Production Memory. The problem of the creation of the $P_{i,j}$ and the R_k is analogous, though additional complexities come in for the $P_{i,j}$ (which must merge two productions, X1 and X2).

This view essentially makes learning occur out of the attempts of the cognitive system to cope with language by means of deliberate action (though it says nothing of what relation that deliberate action bears to the ability to articulate what is going on). Thus, the occasions for learning arise whenever the X-productions are engaged in coping with some language behavior that PS.Harpy knows but hasn't already assimilated into the net.

This analysis pushes the questions of sufficiency back to where the knowledge comes from that gets into WM. In major part this question is no longer concerned with PS.Harpy and the peculiar network characteristics it incorporates. Thus we can safely leave it. One question that does remain is whether the assumption about knowing the state symbols poses any special difficulties. We dealt with this briefly in issue (S5), and have little more to say about it here, except to note that it could cause genuine problems.

Summary on sufficiency analysis. We have looked at the central issues in whether PS.Harpy is ruled out of court on sufficiency grounds, ie, because it uses mechanism that are obviously beyond human capabilities. As might be expected, the analysis was at some points inconclusive. The closest we came to outright rejection were in the numbers of productions, etc. Projections on how Harpy-like schemes might improve brought the down to the merely huge. Still, the size of the net must remain a point of genuine vulnerability for a Harpy-like model. The other major direction for vulnerability, the fixed net, showed itself to be somewhat porous on inspection, in that mechanisms could be found for approaching the various issues. But the analyses were fragmentary and far from satisfactory. For myself, PS.Harpy seems not rejectable on sufficiency grounds, but the reader will have to make his own judgment.

7. SOME SPEECH PHENOMENA

PS.Harpy, being a theory of speech perception embedded in a theory of general cognition, should make a prediction (right or wrong) on almost any speech phenomenon. This theoretically pleasant state of affairs is thwarted in several ways: PS.Harpy is still incomplete; a prediction can depend on the content of productions that we don't know in enough detail; we are unable to extract the predication from the system; or, contra assumption, the phenomenon to be predicted really does lie outside the domain proper of PS.Harpy. Let us discuss a few phenomena of speech perception to see what position, if any, PS.Harpy takes and how inferences occur. Detailed treatment is outside the scope of this paper. In the following we do not always distinguish Harpy from PS.Harpy, though sometimes predictions would hold for any implementation of Harpy and sometimes only for PS.Harpy.

These implications of PS.Harpy are drawn informally, and in two distinct ways. First, assertions are made about the behavior that will occur from PS.Harpy (sometimes in conjunction with other productions). This inference should be, but is not, demonstrated either formally (which is hard) or from running programs. Second, there are almost always multiple ways to do a task within a programming system, ie, within an architecture such as HPSA77. The assertion that a given method (ie, production system) is used requires showing that other potential methods are not used. In general this is very hard to do, since it means considering the class of all methods for doing a given task. One purpose of the rather explicit derivation of the mapping in Section 4 was to indicate where alternative productions systems did or did not seem possible. In the material below we rarely go even this far.

(P1) Automatic extrapolation of speech. A common phenomena is that the listener automatically predicts the speaker. The conversational form of this is finishing sentences for speakers. A basis for this phenomena is shown explicitly in the Shannon guessing game and implicitly (presumably) in all the results that show better memory as a function of increased constraint (approximation to meaningful, grammatical discourse) (Miller & Selfridge, 1953). The phenomena is stronger than the experimental results, because it says the hearer does such prediction en passant while listening and not as if it were an additional task.

PS.Harpy clearly shows a tendency to do this in the $P_{i,j}$ productions, which extrapolate forward without the speech input. This is not quite enough, however, since the actual prediction of the utterance within PS.Harpy requires that the backtrace occur. As we noted, Harpy adopts relatively artificial conditions for backtrace initiation (the end of the total utterance). If we assume, consistent with U1, that one criteria is silence (ie, a pause), then the backtrace will be initiated whenever the speaker pauses unduly. The extrapolated speech is as much a candidate for the maximum as actually heard speech, so that automatic prediction will occur over pauses, etc. Whether PS.Harpy will run ahead of the speaker will depend on the speaker's rate, since PS.Harpy's rate is determined by its internal cycle time.

The extrapolated states undergo decay. Hence, an important question is whether PS.Harpy will extrapolate for long enough to produce the observed phenomena. In general, only a few cycles of extrapolation seem possible or the beam will become too diluted with

unsupported paths. Recall that Harpy operates to assure only about 4 cycles, which is approximately 200 ms., or about 1 word's worth. This seems not enough. However, a closer look reveals that the decay rate is set to eliminate paths in competition with heard speech, which is getting boosted. When there is no input, so that all paths are extrapolations, then the duration will be as long as state elements will win in competition with non-speech activities. Furthermore, if the decrements are governed by a priori confidence (the $k_{i,j}$ of (6.1)), then the maximum likelihood path will emerge. We do not have enough information to make quantitative predictions about how long it will last. From the precision estimates of 8 bits, we can get an upper bound of 256 cycles, which is of the order of 5 secs; this at least might permit a few words (a second of speech). Note that the amount of speech predicted will depend on how predictable (ie, constrained) the speech is. When there are a few highly probable alternatives, the $k_{i,j}$ will be small, the decay will be slight, and the path will survive a long time.

(P2) Phonemic Restoration Effect. A strong point of PS.Harpy should be the penetration of cognition into perception -- having moved perceptual processing relatively far into the cognitive engine. Perhaps the best evidence for such cognitive penetration is the phonemic restoration effect (Warren, 1970; see Warren, 1976, for a review). If a noise occludes a phoneme (which in the experimental verifications is actually not there at all) in an utterance where higher linguistic context dictates the identity of the phoneme, then the phoneme will actually be heard, ie, it cannot be distinguished from the other phonemes in the utterance. The occluding noise itself is not precisely localizable; in particular, knowing the experimental arrangement (that the noise filled a gap with no actual phoneme present) does not help to identify which phoneme was restored.

PS.Harpy will exhibit the phonemic restoration effect. The $P_{i,j}$ generate state elements that pass over a noise. The sequence of state elements that it produces in no way reveals what phoneme is not present, ie, that no acoustic support occurred for the gap. We have not presented productions for performing the task of phoneme (or sound) identification, but no basis would exist for making the discrimination.

The question of time localization and order judgments is a large one that we cannot enter into here. However, it is apparent that in PS.Harpy the only immediate basis for time and order judgments is activation, and the only basis for remembering order and localization beyond the duration of initial activation is integration with other elements, ie, integration into the web of experience. Thus the sounds in an utterance are heard physically in order, which results in their being activated in order; but they are remembered in order because of their integration into the state-element sequence of the understood utterance. It follows that the occluding noise (buzz or click) cannot be localized with respect to the utterance because it is not integrated into the utterance, just as is required by the phoneme restoration effect.

Why then should a gap of silence be readily be both detected and localized, which is the case? There are two possible answers within PS.Harpy. One is that prolonged silence (a gap) is the cue in U1 for initiating the backtrace. Hence, not only is a gap universally detected (ie, independent of the net), but its location is well marked. According to this

mechanism, silence gaps will be especially distinguished with respect to their detection and localization. The more general possibility is that any speech sound (as opposed to a buzz or click) may be integrated into the net and hence be perceived and localized. Silence is a speech sound (it generates a (Data ...) element), and hence would be treated this way. This is not enough: the speech sound may not be part of the extended beam at the point of occurrence; furthermore, the path incorporating the sound must be retrieved by the backtrace. This second possibility is less sharp than the first. We raise it because one must find in the system all the different ways in which a phenomena could be generated, not just the more obvious ways.

If we ask what would allow the noise to be localized (and of course it is localized phenomenally within some interval), it would happen when the system could create a structure that encodes the noise relative to other events. For PS.Harpy this means firing a production on it (which is the only way to encode it). But more specifically it means firing a variable-containing production, since no variable-free production can link two items that are contingent with respect to one another. (Ie, to be variable-free is to involve only pre-established items.) Thus, localization cannot occur concurrently when the noise occurs. The noise gets recognized (by the equivalent of R_k productions), but simply sits in WM waiting for some variable-containing production to attend to it. One would expect this to happen at some break points in the processing, namely when the backtrace has a chance to enter. Thus we are brought again to the criteria for initiating the backtrace (U1), which have not been completely spelled out. But it is not unreasonable to consider that these are sometimes at grammatical boundaries etc., as has been found (Fodor & Bever, 1965).

Could PS.Harpy be modified so that extrapolations could be distinguished from interpretations of actual input? This cannot be done in the $P_{i,j}$ since at the time of deposition it is not known which case the state element will belong to. However, it might be done in the Q_i, which could replace the state-element with one that was marked as boosted by acoustic input. This would still be a variable-free production. There will now be, on occasion, pairs of state elements some marked, some not. The backtrace would have to weave its way among these, but this does not seem difficult to accomplish. From this one would have to predict that a person could learn to localize missing phonemes, providing a training regime, analogous to original language learning, could be devised to always maintain the discriminations. The incompleteness of HPSA77 with respect to learning new productions, and especially learning new variable-free productions, does not permit going any further with this problem here.

(P3) The absence of active parsing. It is a feature of Harpy, inherited by PS.Harpy, that there is no parsing stage. Recognition happens in a single (forth and back) pass and when it is done the hearer understands both the utterance and its grammatical and semantic structure. No amount of self observation on the part of the hearer will yield any insight into how the grammatical structure is achieved. All this is in accord with phenomenological observations, at least to first order.

PS.Harpy predicts that there are limits to this automatic aspect of parsing, namely,

what is in the Harpy net. As mentioned earlier, this will not consist of the entire grammar, but will be built up to provide the discrimination to recognize speech. That is, as long as useful constraint exists for recognition, the "compiled" grammar net will grow. But it will not grow in terms of any criteria of grammatical or semantic completeness. Thus in complex novel constructions, such as deeply embedded sentences, we would expect there to be recognition of the short phrases but the cognitive unraveling of the larger structure.

With this result, which seems to be substantially correct as a first approximation, goes a series of potential problems to explain reaction time differences to understand various types of sentence. One recalls the earlier attempts to determine experimentally times for transformations (Miller & Isard, 1963). Though discredited in their original aim, such results could pose a problem.

(P4) Ambiguous utterances. As Harpy stands (also PS.Harpy), a single best utterance is selected by the backtrace. How then does ambiguity occur? Observe that PS.Harpy always generates many alternative paths. Many of these will be the same path through the grammar net, and thus will be equivalent. But variant readings of an utterance that are good matches to the acoustic utterance will also exist in the WM. Whether PS.Harpy becomes aware of them, in terms of being able to engage in further processing, depends on whether the backtrace will pick up more than one of the paths. There would seem to be no difficulty creating such alternative production systems. Thus, the detection of ambiguity is dictated in part by the cognitive set to see the multiple readings. At the current level of specification of PS.Harpy, we have no way of determining whether the standard backtrace for the human is the one Harpy uses (finding only the one best) or is some variant that will see multiple solutions if they are close enough to the best case.

With a large beam many readings would be possible. A human is only able, at best, to get a few readings. Why can't PS.Harpy read out many readings, thus showing a capability well beyond that of humans? First, though the beam is full of alternative paths, we don't have good statistics on how many of them are alternative readings at the grammar-net level, as opposed to alternative readings about how the speaker sounded. Assuming that many grammatical readings are latent in the final beam, PS.Harpy would suffer from severe response interference in trying to read them out. While the backtrace was obtaining one, the others would all be decaying and the decay would be worse the more readings were attempted.

Detecting ambiguity when it occurs takes time in PS.Harpy, just as it does in humans. The extra time is due to the extra trip by the backtrace productions, not to any extra forward-wave processing. In particular, there is no need to reprocess the sentence to "become aware" of the alternative readings. The extra time exists only when the ambiguity occurs; it is not an effect that arises because of the possibility of ambiguity, ie, not an ensemble effect. It would surely seem possible to induce a person to take extra time because of the possibility of ambiguity, but PS.Harpy would have to explain this as being due to a superimposed cognitive stategy for extra processing of the utterance.

The ambiguity is picked up at the point of backtrace, which may not necessarily be at the end of the utterance. Thus we would expect any temporal disruption to occur just a little after the point of ambiguity became clear. In garden path sentences, for instance, no extra time would be expected until the end of the path had been reached.

(P5) Categorical perception. The phenomena of categorical perception has played an important role in the psychology of speech perception (see Studdert-Kennedy, 1976, for one recent overview). Its defining generalization is that perception of speech sounds seems to be categorical, in that discrimination appears to occur by identification (ie, labeling the sound as being in a predefined class), rather than by measurement of differences along acoustic attributes. Importantly, consonants show categorical perception much more than do vowels. It is also now known that categorical perception occurs for domains other than speech and possibly even for non-humans. Thus it appears that the phenomena of categorical perception is losing its status as a key phenomena in understanding speech perception.

However, it is still useful to see what PS.Harpy has to say about the phenomena. Given a speech context, the attempt to compare two sounds (as in the so called ABX paradigm) leads fundamentally to the identification of state elements and their associated grammar nodes, ie, to categorical perception. Without a speech context the sounds must be encoded according to some different encoding structure, eg, chirps, buzzes, etc., and there would not necessarily be categorical perception (as is is what holds for humans). Thus, PS.Harpy shows the basic phenomena.

Why then would vowels be different? Vowels and consonants are not differentiated within PS.Harpy, at least at the level of formulation developed so far. One possible mechanism stems from the vowels being longer. Such a difference, in itself, does not explain anything. But if the categorical processing of vowels were handled in a short time, then time is available for processing the vowels according to other encodings, eg, as continuous sounds. This agrees with several indications that degrading and shortening vowels makes their perception more categorical. PS.Harpy takes no longer to process a vowel than a consonant, considering only the processing from the R_k level up. Normally, in running speech, the extra time from the long segments simply contributes to the total time available to process the entire forward sweep. In speech-sound discrimination experiments, which consist of short single-syllable words, the extra time is available while the sound is still being heard to analyze it as a non-speech sound.

From PS.Harpy's viewpoint whatever phenomena is shown by categorical perception is not unique to speech. Though we have not explored the matter carefully here, nothing in the PS.Harpy scheme is specific to speech. Rather, the presuppositions involve a time-sequenced input to which attention can be devoted continuously, the existence of strong structure (the grammar net), and the existence of stable structure and long learning times (to get the variable-free productions built). But insofar as other domains share these properties, they too can exhibit categorical perception in the same way.

(P6) Word recognition versus letter recognition. The basic phenomenon is that

reaction time to recognize a letter within a word is longer than to recognize the word itself (Savin & Bever, 1970). This is counter intuitive on the grounds of a hierarchical recognition process, which should require the components to be recognized before the higher unit. The phenomena has been shown repeatedly by now and an attempt has been made to generalize it as the recognition of any task-defined unit is faster than recognition of its components (McNeill & Lindig, 1973), which not only makes it apply at all levels (eg, phrases vs words, words vs syllables, syllables vs phonemes, etc), but also gives it a short term character in terms of momentary task set. There is some controversy about this latter extension (Warren, 1976).

How will PS.Harpy perform in such tasks? The forward sweep proceeds independent of concern with what task is to be performed. In some sense it recognizes the whole utterance. There certainly is no staging above the segment level (the (Data ...)) at which components become available. However, the task is not complete until the backtrace extracts and identifies the desired information. With the standard backtrace we would expect the utterance to be identified and the components to be extracted from it, ie, by decoding the word or whatever. What prevents this result from propagating upward indefinitely (eg, the paragraph is recognized before its component sentences) is that the backtrace is triggered on relatively short utterance durations. Whenever the backtrace operates, there will the components (what is identified by the backtrace) be identified before their superordinate units. From what has been said earlier about the criteria for inclusion in the recognition net, it should be apparent that no simple structural property determines when the backtrace operates, but rather a conglomeration of conditions that can be summed up by the phrase "the demands of recognition". So one should not expect to find neat divisions of units from components according to recognition times.

The structure of PS.Harpy permits some prediction of whether results quite different from the reported ones might be found. The backtrace would seem to be rather easily modifiable. What could be achieved by a different backtrace? It might certainly attempt to pick up a feature of the word directly, rather than afterward. It would seem difficult by this means to have the component become faster than the higher unit, but something approaching parity might be attained. Given extensive learning in a highly stable situation (eg, of the kinds demonstrated by Neisser in his search task (Neisser, Novick & Lazar, 1963) and by Shiffrin and Schneider (1977)) one could produce limited inversions of the effect. Though we do not have yet a model for such learning, one would have to predict that an "automatic" response could be learned that would tap onto the existence of the component as it emerged in the forward-sweep.

(P7) The word-frequency effect. The basic phenomena is that familiar words are recognized more easily (accurately in noise, quickly) than rare words (see Gregg, 1976 for a general overview). In word identification in noise the errors are similar to the presented stimulus word. Sometimes the frequency level of the errors is independent of the frequency level of the presented word (Broadbent, 1967); sometimes it seems to reflect its frequency level (Frederiksen, 1971). When the experiment defines small closed populations of words (as opposed to the full vocabulary of the subject), the word frequency effect essentially disappears.

Consider first how long it takes PS.Harpy to respond to a word in an isolated-word recognition experiment. As it stands, the forward wave will travel with the word and then the backtrace will pick out the identification. With only a single word involved, the states in the final beam could hold the word identification directly. In discussing the backtracing component, we had no basis for deciding which states should be able to evoke the grammar node, hence the word identity; it is surely reasonable to assume that boundary states lead back to the word. Hence, the backtrace operates in a single step and the time is determined by the forward wave. Thus, in the standard program there would seem to be no differential reaction-time effect of word frequency.

This leaves the source of the effect in some special feature of the task, namely speeded word identification. There seem to be two possibilities: (1) the backtrace can be initiated sooner; (2) the forward sweep can go faster. In any case, the differential effect must arise from something that is a function (up to correlation) of word frequency. Within the current PS.Harpy, only the $k_{i,j}$ in the $P_{i,j}$ extrapolation productions is a possible source. However, systematic structural differences in words as a function of frequency is also a possibility (Landauer & Streeter, 1971).

Consider triggering the backtrace early. There is no way for the system to examine the whole beam en passant to decide when to do this. However, a fixed criteria could be used. That is, if a path came up above a level determined by an examining production, then the system would initiate finding the word and responding. If the $k_{i,j}$ were a function of frequency then the high frequency words would be more likely to satisfy this criteria, hence to be recognized early.

The errors that occur in such speeded trials would follow the a priori distribution of errors in the beam at some early time. There would not seem to be any reason why the frequency level of the true candidate would affect the frequency-level of the error that was made. This is essentially what Broadbent found. On the other hand, the errors that are made will be similar to the input signal, since they are selected out of the beam that is built in response to the signal, again as found.

Consider the alternative of speeding up the forward sweep. This is composed entirely of a sequence of P-Q pairs, paced by the concurrent firings of R_K to encode the signal. The P-Q pairs form a serial sequence. Without creating a different algorithm there seems no way to remove a $P_{i,j}$, which is the generative step. However Q's are not required. It may be possible for the P's to race ahead to obtain a complete path. Assuming this would lead to triggering the backtrace early, this would lead to speeded recognition. This would happen, if at all, only on high frequency words (with slow $k_{i,j}$ decay), hence producing the differential effect.

Consider next the accuracy of responding under noise. To explore this fully requires extending the model back to the genesis of the R_K to discover how noise can enter into the processing -- another excursion we cannot afford. One simple view says that noise in general leads to a failure of the (Data _) to make contact at all, rather than supporting strongly a

false path. Thus recognitions in noise will have relatively heavy contributions from unsupported extrapolations by the $P_{i,j}$. This returns us to the dependency for such extrapolations on the $k_{i,j}$, so that if they are an appropriate function of frequency (as seems natural), then there is a basis for the word frequency effect.

The result of Pollack et al (1959) that, if the population is explicitly given, then the (basal) word frequency does not count needs also to be explained. These results involve the threshold for detection. This is not the situation in which the modified executive, described above, operates. Rather substantial time is available to make use of the short-term knowledge of the vocabulary. Operating simply as a test to reject candidates suggested by the backtrace already seems sufficient to produce a strong counteraction to the word-frequency effect. Additional shaping of the forward wave may also be possible, in the manner briefly discussed under issue (S5).

(P8) The unit of speech perception. Much of the work in speech perception has revolved around the attempt to discover the "unit", ie, whether recognition is organized around the acoustic segment, the feature, the phoneme, the syllable, or the word. Some of the phenomena already discussed (eg, the word/letter recognition effect) have been seen as parts of this larger theoretical question. PS.Harpy, by providing a complete set of mechanisms for recognition, provides in some sense a complete answer to this question. The (Data ...), which are acoustic segments of variable duration and learned character certainly exist. At recognition time, the state structure does not have any hierarchical organization, so no higher units exist. However, there is a structure from which this state network is derived. This does have hierarchical structure, hence units. In Harpy this is a two level affair, the lexical and the grammatical level. But this part of the organization of PS.Harpy could be different in many ways and still make little difference to the operational structure of (6.1) - (6.8).

In any event, it should be realized that the question of the unit of speech perception becomes slightly anachronistic once complete computational organizations for perception are proposed. They invariably have a more complex reality, which then becomes the central object of scientific interest.

8. CONCLUSION

We have now achieved our goal: We have produced a theory of human speech perception based on Harpy that is sufficient to perceive connected speech and is not implausible in terms of human information processing capabilities.

But what is it that have achieved in fact? There are several aspects that need to be discussed.

Sufficiency. Harpy (or, better, Harpy-extended) is not known to be sufficient to recognize full human speech. This is an important limitation of the results obtained here. However, it is not basically of concern to me. Harpy's current and potential power are sufficient unto the day thereof. As we learn more about how to recognize speech, Harpy may prove to be a technical dead-end and other mechanisms, say more akin to Hearsay-II, will emerge. That will be all right with me.

More important, the projection of Harpy into PS.Harpy has produced estimates for the complexity of the computation: how many productions, how much computation, and how much working memory. These numbers seem uncomfortably large, though not out of the question with the improvements that might be possible in a Harpy-like technology (eg, in the depth, D, the beam fraction, f, and the compression, c).

It is possible to conclude from these numbers that Harpy is not a good basis for human speech perception. However, such a conclusion should produce an uncomfortable feeling. For there is no system around (and no design for one) that takes any less computation. Indeed, to my knowledge, Harpy is the best scheme computationally (in terms of millions of instructions per second of speech). And PS.Harpy already exploits to the extreme that great computational loophole, parallel computation. Nothing comes for free, of course, so what PS.Harpy has purchased in time it has paid for in hardware (the large number of productions). I would hope that those who reject PS.Harpy as a theory on computational grounds would discuss the computational alternatives in some quantitative detail.

We also do not have a complete picture of the information processing limits of the human. Some numbers are reasonably large (eg, neuron populations, which get above 10^8 with no difficulty); some numbers are pretty small (eg, the symbolic computation cycle is highly unlikely to be less than 20 ms). So it is hard to know when numbers are out of range. However, the biggest gap in our knowledge would seem to be about the structural limits on the nature of learning -- on what form things must take in order to be learned. Here our analysis is completely inadequate.

PS.Harpy as a psychological theory. Sufficiency analysis is only one item of evidence. The psychological literature is replete with speech-perception related phenomena, all of which can serve to test a theory and refine it further. We discussed a small collection of such phenomena. Our analysis dealt only with the basic findings and at a qualitative level. PS.Harpy exhibits many of the right characteristics, though difficulties could crop up on any of them as the theory attempts to make more thorough and detailed predictions.

PS.Harpy was not defined with any of these specific phenomena in mind. Thus, in a subjective sense, all of these predictions count for the theory. However, as noted, the informal derivations were suspect in a couple of ways. In particular, demonstrating that PS.Harpy must show a given behavior in a task requires considering the class of all methods for the task that can be devised within the architecture. Other methods always exist, and it must be shown that they would not be learned or would not be selected on this occasion, etc. Without such analysis, there seems too high a chance of selecting out the one method that has the desired properties. That such a (desired) method exists is important, but it is only half the argument. Thus, I consider the analyses in Section 6 simply heuristic. Primarily, they make the case that PS.Harpy is a serious contender as a psychological theory. I don't wish to underplay the extent to which PS.Harpy appears to have the right properties. I consider the indications in Section 6 remarkable, especially since they were in essence simply "read off" the existing system.

On HPSA77. It is regrettable that HPSA77 had to be introduced without more analysis and support. Nevertheless, PS.Harpy, now that it is complete, stands on its own as a plausible theory. Its could have been proposed de novo, just from the notion of a basic production system architecture, such as has been current in psychology for some time. Assumption D3, on the activation of productions, probably would have dropped out, as it has played little role in the present analysis.

What was gained in mapping Harpy into HPSA77, rather than just considering the psychological implications of Harpy directly? We addressed the question briefly at the beginning, but the issues can be seen more clearly now. On the positive side, the psychological interpretation of HPSA77 is well tied down. This shows in the ability to make time predictions for speech based on behavior in memory experiments. The prediction was not completely tight here, because the timing specifications have not yet been made for HPSA77; but the tie between these two classes of phenomena is quite clear. If we had not had HPSA77, or some equivalent, we would have continually found ourselves with open fields of possibilities for how to realize the features of Harpy in the human. Our plight would have been much more like the diagram presented in Klatt's paper (these proceedings), which, whatever its merits, is relatively free in introducing functional boxes. Finally, we actually ended up with an explicit model for speech perception, (6.1) - (6.8), which would have been impossible without some such constraint as HPSA77 provides. Of course, if the model is basically wrong, then such minor virtues are little consolation. One the negative side, many of the explanations of speech phenomena depended only on the features of Harpy and not how these were implemented. Here, use of HPSA77 may obscure the essential structure of the explanation, though the more explicit theoretical structure might make it easier to derive

Critical weaknesses of PS.Harpy. The analysis dealt rather completely with the performance of perception. The most vulnerable aspect of the performance system is the backtrace. In part this is the (apparent) clash with the design decision D3 on positing activation on productions so that experiencing in the forward direction was possible. No specific difficulties were uncovered, but philosophically it seems wrong to have the speech system routinely working backwards through the utterance, when the human has difficulties

doing it for deliberately held memory-span tests. The backtrace productions are variable-containing productions, entirely analogous to the performance productions that would exist for the memory-span task. One possibility is that the linked structure of the analyzed speech utterance, which does not exist for the typical memory-span task, makes a critical difference.

There are other issues with the backtrace as well. It operates like an executive routine and variations in it were proposed freely in the discussing psychological phenomena. A more thorough analysis is required to understand what is possible and what is not. This is place where the lack of an analysis of the entire class of methods produces a sense of unease. This same freedom implies that there may be schemes for doing the backtrace that work forward in various ways, or that do parts of the task concurrently with the forward sweep. Such variations, were they to exist, might change some of the conclusions of the paper.

The second major deficiency in analysis of PS.Harpy is the lack of detail on the acoustic side. Having jettisoned the Itakura metric used in Harpy as a candidate for detailed mapping, it was left open what sort of metric would do. It was not appropriate to invent a new metric in this paper. This had the effect of truncating the entire acoustical analysis.

Related to this, the functional boundary between the cognitive system and a distinct sensory-perceptual system remains open, though the theory localizes it firmly in WM. Two principles help determine this boundary. First, any effect of cognition on perception must occur in WM; this forces various perceptual products to be symbols in WM. Second, anything that happens in WM is open to cognitive manipulation and modification by the acquisition of new productions; this forces various perceptual products to reside outside of WM. Beyond this, there is no implication about how much specialized perceptual processing might happen. That PS.Harpy stopped with the R_k is of little moment. If we were willing to assume the (Data ...), we could have jettisoned the R_k; similarly, prior stages of chunking and processing by productions could have occurred in WM to produce the symbols taken as input by the R_k. What data would help define this boundary isn't evident. The Phoneme Restoration Effect, perhaps the most obvious candidate, was explained already within PS.Harpy. In any event, we were unable to provide a serious assessment of the adequacy of the recognition match to handle the measure of closeness between the template and the acoustic parametric signal. With this lack, PS.Harpy has to remain silent on many speech phenomena.

There was almost no analysis on the details of the process of compilation and how PS.Harpy would learn the network. With this went a loss of plausibility constraint on the forms of the productions. Could they have been learned? For instance, could the $k_{i,j}$ (in the $P_{i,j}$ productions) be learned? The failure here stems in part from HPSA77 not having adopted yet any design decision with respect to learning new productions. Why this is the case is too extended a story for this paper. But also in part the failure stems simply from the learning context opening up the analysis to new tasks and new contexts, which would have multiplied the effort of this paper many fold. Insofar as the concerns about Harpy rest on its fixed, compiled nets, then the analysis of this paper has only set the stage for the real sufficiency analysis still to come.

Relation to other theories. We limit ourselves to only a couple of remarks about other theories. First, with respect to Dennis Klatt's effort, being developed and presented concurrently with this (Klatt, this proceedings), my general debt has already been expressed. His tactic, which is to draw on a range of new mechanisms, to compose what seems like an appropriate total scheme, seems eminently sensible to me -- in some sense a better tactic that my own, which accepts one system in toto. His theory provides a clearer picture of some of the essentials for adding new words, etc., aspects that were left hanging in PS.Harpy. One the other hand, he posits functional boxes freely around his system, compared with PS.Harpy, which really does fit almost everything within a known control structure. Furthermore, he weakens the sufficiency aspects of his system considerably, since it now is simply a new design for a Speech Understanding System.

PS.Harpy is not a motor theory. That is clear. The sufficiency of Harpy shows that analysis by synthesis is not necessary at the point of performance -- though limitations on Harpy's extendibility could show that demonstration to be in error. Given that PS.Harpy is both sufficient and plausible psychologically, the generalized sufficiency-like arguments behind both the motor theory and analysis-by-synthesis simply lose their force. What might be true at the learning level is an entirely different matter, though neither of these motor theories make use of learning arguments. Likewise, whether there is a role at the cognitive level for speech production involvement in the recognition of difficult speech is also an open question.

Extension to the rest of perception. We have concentrated here on speech perception, though we took a small excursion in Section 5 part way into psychophysics in order to ground a given design decision. Having specified HPSA77 this far (ie, to PS.Harpy), our treatment of other senses is highly constrained. While it is possible that vision, for example, could be handled entirely differently, the main line approach would certainly be simply to include the visual sensory buffer (the icon) as part of WM (already assumed in D4), and to do the visual recognition in essentially the same manner as we have done speech, taking into account the different structural features of the task (eg, two dimensions, size, knowledge structure, etc). Such an undertaking might not be quixotic at all. Technologically, the group around Raj Reddy at CMU are following exactly this path. A recent thesis (Rubin, 1978) has shown what has to be done to adapt Harpy techniques to natural scene analysis and has demonstrated its basic feasibility.

Methodological notes. Some comments on the methodological aspects of this study are in order.

The first is the fruitfulness of sufficiency analysis. This paper was meant to demonstrate the usefulness of this technique. I am satisfied with the exercise in this respect and hope others are induced to consider sufficiency issues more intensively.

The second is the power of the mapping. Starting with Harpy and an architecture, a full blown psychological theory of speech perception was manufactured in short order. I can testify personally to the power of this device, since the entire theory was force generated in

a few days, starting with just the mapping assumptions (M1) - (M5), and unwinding essentially as it appears in the paper.

The third is the benefit of having a theory of the architecture of human cognition. The argument needs to be made more generally in connection with HPSA77; but this paper seems to me a data point about how useful such a construct is. Of course, it pays to have the *right* architecture, but I wish to note explicitly the power an architecture provides for making scientific progress. HPSA77 is a design probe and, although it is wearing rather well at the moment, it cannot possibly survive the attempt to extend it to cover all of cognition. But it seems to be a good vehicle.

The fourth is how a complete model of cognition has something to say about a very wide range of phenomena. PS.Harpy offers explanations on many phenomena, not because it was specifically designed to cover them, and not because additional assumptions are added each time, but because it was designed to be a full system for the perception of speech, hence must exhibit behavior in all speech situations.

The fifth is the role of minimally specified theories to keep design decisions open to accommodate new results in new task domains. This runs counter to some popular notions about rushing headlong for falsifiability. It also seems as if the theory were being adapted post hoc to meet each new result. But it seems a remarkably sensible strategy when a complex system such as the human brain has to be modeled. Far from being post hoc, there is an accumulation of design specification (ie, of theoretical constraint) with each act of accommodation. As long as the theory has sufficient precision, all the chickens will come home to roost eventually. This tactic does lead to the temporary inability to predict behavior in various tasks, because of underspecification. PS.Harpy shows this in several ways. But this should just be seen as selecting an appropriate approximating sequence for developing a theory. After all, the empirical results that will cause difficulty are patient -- they will lie in wait for a very long time.

REFERENCES

Anderson, J. R. *Language and Thought.* Erlbaum, 1975.

Anderson, J. R., Kline, P. J. and Beasley, C. M. Theory of the Acquisition of Cognitive Skills. ONR Technical Report 77-1, Yale University, 1977.

Anzai, Y. Learning strategies by computer, *Proceedings of the Second National Conference,* Canadian Society for the Computational Study of Intelligence, University of Toronto, Canada, 19-21 July 1978.

Baker, J. The Dragon System: An Overview, *IEEE Trans. Acoustics, Speech and Signal Processing ASSP-23,* 24-29, 1975.

Baylor, G. and Gascon, J. An information processing theory of aspects of the development of weight seriation in children, *Cognitive Psychology 6,* 1-40, 1974.

Broadbent, D. Word frequency effect and response bias, *Psychological Review 74,* 1-15, 1967.

Cole, R. and Jakimik, J. A model of speech perception. This proceedings, 1978.

Cole, R., Rudnicky, A., Reddy, R. and Zue, V. Speech as patterns on paper. This proceedings, 1978.

Crowder, R. G. *Principles of Learning and Memory,* Erlbaum, 1976.

Erman, L. and Lesser, V. A retrospective view of the Hearsay-II architecture. *Proc. Fifth International Joint Conference on Artificial Intelligence,* MIT, 790-800, 1977.

Estes, W. K., An associative basis for coding and organization in memory, in A. W. Melton and E. Martin (eds.) *Coding Processes in Human Memory,* Winston, 1972.

Fodor, J. A. and Bever, T. G. The psychological reality of linguistic segments, *J. Verbal Learning and Verbal Behavior 4,* 414-420, 1965.

Forgy, C. and McDermott, J. OPS2: A domain-independent production system language. *Proc. Fifth International Joint Conference on Artificial Intelligence,* MIT, 933-939, 1977.

Frederiksen, J. R. Statistical decision model for auditory word recognition, *Psychological Review 78,* 409-419, 1971.

Goldberg, H. G. Segmentation and Labeling of Speech. PhD Thesis. Department of Computer Science, Carnegie-Mellon University, 1975.

Green, D. M. and Luce, R. D. Speed-accuracy trade off in auditory detection, in S. Kornblum (ed.), *Attention and Performance IV*, Academic, 547-569, 1975.

Gregg, V. Word frequency, recognition and recall, in J. Brown (ed.), *Recall and Recognition*, Wiley, 1976.

Halle, M. and Stevens, K. N. Speech recognition: A model and a program for research. *IRE Trans. Information Theory IT-8*, 155-159, 1962.

Hunt, E. B. and Poltrock, S. E. The mechanics of thought, in B. H. Kantowitz (ed.) *Human Information Processing: Tutorials in Performance and Cognition*, Erlbaum, 1974.

Jelenek, F. Continuous speech recognition by statistical methods. *Proc. IEEE 64*, 532-556, 1976.

Klahr, D. and Wallace, J. G. *Cognitive Development: An information processing view*, Erlbaum, 1976.

Klatt, D. Review of the ARPA Speech Understanding Project. *J. Acoust. Soc. Am. 62*, 1345-1366, 1977.

Landauer, T. K. and Streeter, L. A. Structural differences between common and rare words: Failure of equivalence assumptions for theories of word recognition, *J. Verbal Learning and Verbal Behavior 12*, 119-131, 1973.

Langley, P. BACON.1: A general discovery system, *Proceedings of the Second National Conference*, Canadian Society for the Computational Study of Intelligence, University of Toronto, Canada, 19-21 July 1978.

Liberman, A. M., Cooper, F. S., Shankweiler, D. S. and Studdert-Kennedy, M. Perception of the speech code. *Psych. Rev., 74*, 431-461, 1967.

Lowerre, B. T. The Harpy Speech Recognition System. PhD Thesis. Department of Computer Science, Carnegie-Mellon University, 1976.

Luce, D, The theory of selective information and some of its applications, in D. Luce (ed.), *Developments in Mathematical Psychology*, The Free Press, 1960.

Luce, D. and Green, D. M. A neural timing theory for response times and the psychophysics of intensity, *Psychological Review 79*, 14-57, 1972.

McCracken, D. A Production System Version of the Hearsay-II Speech Understanding System, PhD Thesis. Department of Computer Science, Carnegie-Mellon University, 1978.

McDermott, J. and Forgy, L. Production system conflict resolution strategies, in D. W.

Waterman and F. Hayes-Roth (eds.). *Pattern-Directed Inference Systems.* Academic Press, 1978.

McGill, W. J. Neural counting mechanisms and energy detection in audition, *J. Mathematical Psychology 4*, 351-376, 1967.

McNeill, D. and Lindig, L. The perceptual reality of phonemes, syllables, words and sentences, *J. Verbal Learning and Verbal Behavior 12*, 419-430, 1973.

Medress, M., et al. Speech Understanding Systems: Report of a Steering Committee. *SIGART Newsletter*, No. 62, pp 4-8, 1976.

Miller, G. A. and Isard, S. Some perceptual consequences of linguistic rules, *J. Verbal Learning and Verbal Behavior 2*, 217-228, 1963.

Miller, G. A. and Selfridge, J. A., Verbal context and the recall of meaningful material, *American J. Psychology 63*, 176-185, 1953.

Moran, T. The Symbolic Imagery Hypothesis: An empirical investigation via Production System Simulation of Human Behavior in a Visualization task. PhD Thesis. Department of Computer Science, Carnegie-Mellon University, 1971.

Morton, J. and Broadbent, D. E. Passive versus active recognition models, or Is your homunculus really necessary? in W. Wathen-Dunn (ed), *Models for the Perception of Speech and Visual Form*, MIT, 103-110, 1967.

Murdock, B. *Human Memory: Theory and data*, Erlbaum, 1974.

Nakagawa, S. A Machine Understanding System for Spoken Japanese Sentences, PhD thesis, Department of Information Science, Kyoto University, 1976.

Neisser, U., Novick, R. and Lazar, R. Searching for ten targets simultaneously, *Perceptual and Motor Skills 17*, 955-961, 1963.

Neves, D. M., A computer program that learns algebraic procedures by examining examples and by working test problems in a text book, *Proceedings of the Second National Conference*, Canadian Society for the Computational Study of Intelligence, University of Toronto, Canada, 19-21 July 1978.

Newell, A. Production systems: Models of control structures, in W. C. Chase (ed.) *Visual Information Processing.* Academic Press, 463-526, 1973.

Newell, A. Knowledge representation aspects of productions systems, *Proc. Fifth International Joint Conference on Artificial Intelligence*, MIT, 987-988, 1977.

Newell, A. A Production System Architecture for Human Cognition, 1978 (In preparation).

Newell, A., et al. *Speech Understanding Systems: Final report of a Study Group.* North-Holland/American Elsevier, 1973. (Reprinting of 1971 study)

Newell, A. and Simon, H. A. *Human Problem Solving.* Prentice-Hall, 1972.

Peterson, L. R. and Peterson, M. J. Short-term retention of individual verbal items, *J. Experimental Psychology 58*, 193-198, 1959.

Pollack, I., Rubenstein, H. and Decker, L. Intelligibility of known and unknown message sets, *J. Acoustical Society of America 31*, 273-279, 1959.

Reddy, R., et al. Speech Understanding Systems: Final Report. Department of Computer Science, Carnegie-Mellon University, 1977.

Ritea, B. Automatic Speech Understanding Systems. *Proc. 11th IEEE Computer Society Conference.* 319-322, 1975.

Rubin, S. The ARGOS Image Understanding System. PhD Thesis. Department of Computer Science, Carnegie-Mellon University, 1978.

Rychener, M. Production Systems as a Programming Language for Artificial Intelligence applications. PhD thesis. Department of Computer Science, Carnegie-Mellon University, 1976.

Rychener, M. and Newell, A. An instructable production system: basic design issues, in D. A. Waterman and F. Hayes-Roth (eds.) *Pattern-Directed Inference Systems.* Academic Press, 1978.

Savin, H. B. and Bever, T. G. The nonperceptual reality of the phoneme, *J. Verbal Learning and Verbal Behavior 9*, 295-302, 1970.

Schneider, W. and Shiffrin, R. M. Controlled and automatic human information processing: I. Detection, search, and attention, *Psychological Review 84*, 1-66, 1977.

Shiffrin, R. M. and Schneider, W. Controlled and automatic human information processing: II. Perceptual Learning, automatic attending, and a general theory, *Psychological Review 84*, 127-190, 1977.

Simon, H. A. The architecture of complexity. *Proc. American Philosophical Society 106*, Dec. 1962.

Simon, H. A. How big is a chunk? *Science 183*, 482-488, 1974.

Slate, D. J. and Atkin, L. R. CHESS 4.5: The Northwestern University chess program, in R. W. Frey (ed.) *Chess Skill in Man and Machine*, Springer-Verlag, 82-118, 1977.

Sternberg, S. The discovery of processing stages: Extensions of Donders' method, in W. G. Koster (ed.), *Attention and Performance II, Acta Psychologia 30*, 276-315, 1969.

Studdert-Kennedy, M. Speech Perception, in N. J. Lass (ed.), *Contemporary Issues in Experimental Phonetics*. Academic, 1976.

Warren, R. Perceptual restoration of missing speech sounds. *Science 167*, 392-393, 1970.

Warren, R. M. Auditory illusions and perceptual processes, in N. J. Lass (ed.), *Contemporary Issues in Experimental Phonetics*. Academic, 389-417, 1976

Waterman, D. A. Adaptive production systems, *Proc. Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, USSR, 296-303, 1975.

Waterman, D. and Hayes-Roth, F. (eds.) *Pattern-Directed Inferences Systems*. Academic Press, 1978.

Woods, W., et al. Speech Understanding Systems: Final Report, Bolt, Beranek and Newman, 1976.

Young, R. Children's Seriation Behavior: A Production System Analysis. PhD thesis. Department of Psychology, Carnegie-Mellon University, 1973.