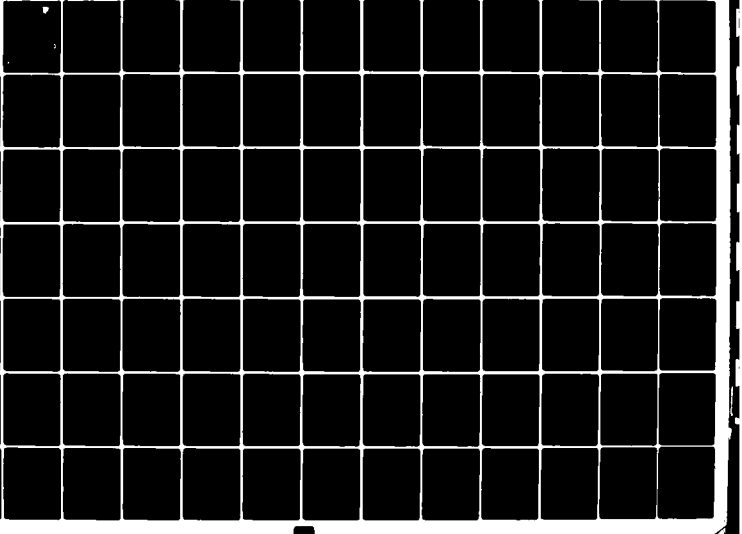
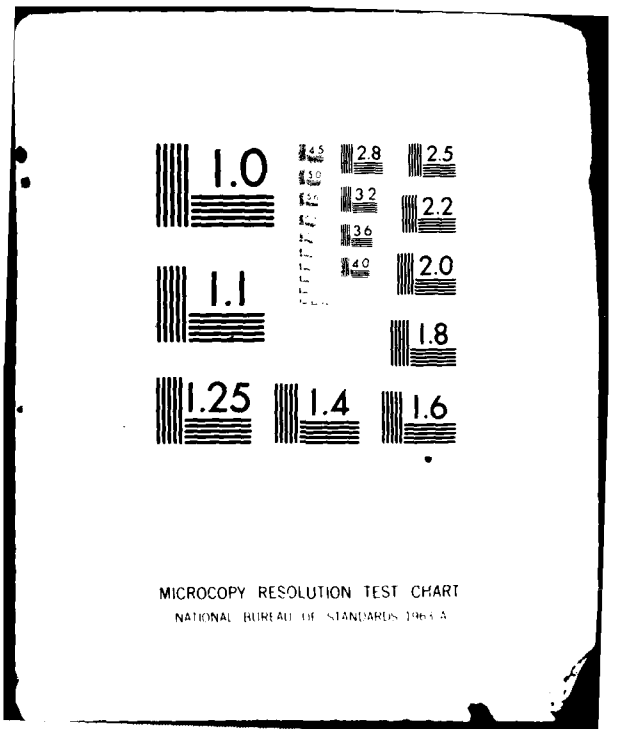
AD-A110 809 SYRACUSE UNIV NY SCHOOL OF COMPUTER AND INFORMATION --ETC F/G 9/2
INTEGRATED PARALLEL PROCESSES: THE ELEMENTS OF MEANING IN LANGU--ETC(U)
NOV 81 E F STORM F30602-77-C-0236

UNCLASSIFIED RADC-TR-80-379-VOL-4 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

LEVEL II

12

AD A110809

RADC-TR-80-379, Vol IV (of five)
Final Technical Report
November 1981

INTEGRATED PARALLEL PROCESSES: THE ELEMENTS OF MEANING IN LANGUAGE

Syracuse University

Edward F. Storm

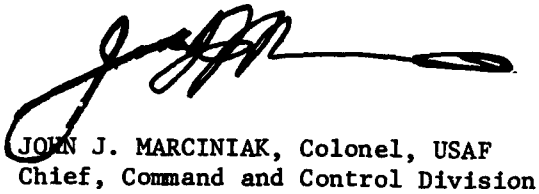
ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-379, Vol IV (of five) has been reviewed and is approved for publication.

APPROVED: 

CLEMENT D. FALZARANO
Project Engineer

APPROVED: 

JOHN J. MARCINIAK, Colonel, USAF
Chief, Command and Control Division

FOR THE COMMANDER: 

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIS) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-80-379, Vol IV (of five)	2. GOVT ACCESSION NO. AD-A110809	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTEGRATED PARALLEL PROCESSES: THE ELEMENTS OF MEANING IN LANGUAGE		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 1 Oct 77 - 30 Sep 80 6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Edward F. Storm		8. CONTRACT OR GRANT NUMBER(s) F30602-77-C-0235
9. PERFORMING ORGANIZATION NAME AND ADDRESS Syracuse University School of Computer & Information Science Syracuse NY 13210		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 55811903
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIS) Griffiss AFB NY 13441		12. REPORT DATE November 1981 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Clement D. Falzarano (CO)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Programming Systems	Cor Modeling
Programming Languages	Sys. . ulation
Programming Grammars	Schedul . Algorithm
Proving Programs Correct	Logic Programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The "Language Studies" contract is divided into four project areas, all of which are directed to the problems of effectively, reliably and efficiently using modern computers in a wide range of applications.

Three of the projects deal with methods of communicating with computers. Task 1. Very High Level Programming Systems (P.I.: J.A. Robinson). This group is working towards combining the features developed to support work in the area of artificial intelligence and those used in general program

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

development into a new conceptual framework that can be understood and used by a large community of users. Task 2. Proving Program Correctness (P.I: J.C. Reynolds). This group is working towards programming language designs which increase the probability that specification errors will be detected by the compiler or interpreter and to provide the language facilities so that users will more nearly be able to prove that programs perform as they are specified than is currently possible. Task 3. Grammars of Programming (P.I.: E.F. Storm). This group is working towards the development of methods which will allow users to communicate with computer programs in terms more normal to their every day communication forms. Task 4. Systems Studies (P.I.: R.G. Sargent). This group is working towards developing more sophisticated and efficient models of computer systems which can predict system performance when given particular parameter values. The current efforts concern models of transaction processing systems (TPS).

Preface

This report describes efforts completed in the Language Studies project at Syracuse University under RADC contract F30602-77-C-0235. The work covers the period October 1, 1977 through September 30, 1980.

The report is produced in five volumes to facilitate single volume distribution.

Volume 1. Report from the Very High Level Programming Systems task. Report title is "Logic Programming in Lisp".

Volume 2. Report from the Systems Studies task. Report title is "Multiple Finite Queueing Model with Fixed Priority Scheduling".

Volume 3. Report from the Systems Studies task. Report title is "An Algorithmic Solution for a Queueing Model of a Computer System with Interactive and Batch Jobs.

Volume 4. Report from the Grammars of Programming task. Report title is "Integrated Parallel Processes: The Elements of Meaning in Language".

Volume 5. Report from the Proving Program Correctness task. Report title is "Proving Program Correctness" task.

Abstract

This document reports in summary form some investigations into the notion of meaning underlying language, thought and behavior in humans. The general conclusion is that a rigorous specification of cyclic or recurring processes, and the hypothesis that these processes are the realities that underly language, thought and behavior will significantly expand our understanding of these issues, and will enhance our capacity to develop disciplined and responsible technology to improve human communication. In particular, we may look forward to more fully adequate and more sensitive systems for communication between the human and the computer.

This report presents its material in a preliminary way and discusses in outline form some ideas current in the relevant literature that led to that material. Precise formulations are not included, nor are there adequate examples to justify the material. Careful and thorough formulation, adequate implementation, and substantive applications constitute the next phases of this project.

Acknowledgements

Conversations with Dr. Robert L. Morris have contributed in fundamental and specific ways toward identifying what is important in human communication, and what must be represented in a formulation. Some of the ideas in this report are his.

I am in a special debt to Mr. Guy Snedeker. He has helped with organizing concepts, identifying new ones, illustrating them and above all, judging their significance.

Contents

0. Summary

In the introduction we review some general phenomena of language and consider some matters of perspective in its study. In the second section we present a skeleton of a theory for the forms found in language, thought and behavior, and for the integration of personality factors, integrative processes which we believe affect language and thought in essential ways. In the third section we discuss some general issues in the relation between syntax (or grammar) and meaning and relate them to the perspectives discussed in the first section.

In the fourth section we consider computation regarded as a physical or real process, and see that the issues that arise clarify the widespread suspicion that the brain/mind (or whatever) is a computer. In the fifth section we consider very briefly the idea that the brain is a computer, and we see that the suspicion is not confirmed by direct observation of brain structures and processes.

In the sixth section we consider the contemporary study of grammar from the point of view of computability theory and see that what is described in computational terms is necessarily void of interpretation - of meaning in terms of familiar concepts and abstractions. In section seven we look at the abstact theory of computing and see what is involved in concluding that a process is a computation, or that a phenomenon is a computed structure.

In the eighth section we review two theories about semantics in generative grammar and see that they are really alternative theories about the organization of computational systems. The ninth section looks superficially at just a few ideas about the structure of mental images, and the tenth section looks in detail at the deep structures of grammatical theory and how they interact with lexical processes. In the eleventh section we survey some ideas about lexical structures themselves.

Section twelve offers an intuitive, not precisely specified, theory of the forms of meaning, a theory that regards those forms as recurrent processes that can combine in specified if not fully definite ways, and can undergo changes in combination dynamically, where the forms of these changes must adhere to rule but may be unpredictable in general. We postulate that all these forms are the "carriers" of experience or perception. In the last section we describe the prototype of a computer system to implement cooperating recurrent processes.

The Appendix summarizes some earlier ideas about the nature of occurrences. Those ideas did not lead to productive results until we appreciated that dynamic processes could be identified exactly with occurrences. These processes can be simulate,. and that simulation is what is described in the thirteenth section.

1. Introduction

"I never forgot it. I went to a seminary just after high school, and we had religion class on Sunday. This one Sunday afternoon something must have gotten to the priest who taught us because he was really fired up. He said that there were, as in most other human systems, two forms of religion. One, he said, was concerned with devotion, with ritual, with keeping up to a set of rules and regulations. This variety, and he spoke with distaste, tried to bring deep and universal philosophical and theological issues down to the level of the common man. But the end result, he went on, was often nothing more than bead jingling and the worship of statues. The other variety, he said, was concerned with the ultimate nature of reality, with the human spirit, with the ultimate conscious source of all reality. It tried to understand our destiny in terms of this ultimate consciousness. This latter variety, he said, seems to be beyond the intellectual capacity of most human beings, but it was the only variety that was intellectually worth studying. Bead jingling, he said, was mere operationalism, a word I did not yet know.

"His remarks brought my mind to a standstill then, and I've never been the same since. Every time I come across a body of thought that proposes to be a systematic treatment of some class of phenomena I ask myself whether I am looking at the deep and substantive side of it or whether I am seeing only the bead jingling. Technology has always struck me as the bead jingling side of science, and I guess that's why I have never been able to get interested in technology for its own sake.

"Only some years later did I come to appreciate that this priest had missed something. I think he did not appreciate that both aspects are needed if you are to have a meaningful religion, and that in general both aspects are needed if you are to have a meaningful system of thought on any subject.

"Maybe he was right in identifying operationalism with bead jingling. But his tone suggested, at least to me, that its distance from theory made it inconsequential, if not foolish. If that's what he meant, then he was wrong for sure. There are surely universal organizing principles that govern all of reality, or at least whatever of it we can know. But we sort these principles out with reference to our statues and beads, and we learn a lot by illuminating our statues and jingling our beads."

This protocol, from a confidential counselling session, occurred naturally in four segments, represented here as paragraphs. Each paragraph in turn consists of a sequence of sentences. Some of these are factual and others are metaphorical. And the subject matter of this discourse is hardly concerned with immediate and tangible aspects of physicial reality. Experience might have provoked it but its topics are quite abstract and remote from such notions as mass, position, momentum, or the shapes, sizes and movements of constellations of physical objects

in a localized environment.

It is an interesting exercise to appreciate that the shift from the first four paragraphs here to the next one was accompanied by what we have to call, for want of a better terminology, a kind of "mental feeling", a feeling that accompanies not a shift from one topic to another, but a shift from one level of attention and organization to another, related but different, level of attention and organization. And in fact this present paragraph itself provokes yet another such shift.

The task that confronts the study of language is to understand how such arrangements of verbal signals as those exhibited above and their associated shifts in impression are implicated with human thoughts and feelings, and in particular with the kinds of thoughts and feelings that seem to be expressed in those particular signals.

This study began as a study in computational linguistics - how can a computing machine be made to do some of the things human beings seem to do with language. As the study progressed, I came to see that there are two ways to pursue answers to the important questions involved. According to one plan we assemble systems which perform as human beings appear to perform. To the extent that these systems are successful we add them to our technological arsenal. And if we wish, we can try to see whether the technology we used to assemble them is reflected in the biological instruments that humans appear to use to similar ends.

According to the other plan we try first to determine what instruments, what structures and processes the human brings to bear on his activities. When we find them, we may then try to incorporate analogous structures and processes into our technology. This study came finally to focus on this latter kind of plan. It came to be concerned with a search for the structures and processes that underly the human use of language.

A number of disciplines are implicated in this search for these structures and processes. Philosophy through the ages, and in more modern times linguistics and psychology have sought them. Religion is deeply involved, and anthropology and sociology are in important ways relevant. The physical sciences have provided one paradigm for the entrainment of theorizing with observation, and mathematics and modern logic have tried to clarify those most fundamental notions for all thinking humans - the ideas of coherence and of consequence.

Yet, within this spectrum of investigations, one finds again the constrast between observation of what occurs, and the observation of the structures and processes that are instrumental. The search for and the study of these instruments is in principle a study of forms - exactly and definitely specified structures and processes, where exactness and definiteness are achieved by expression in the language of mathematics and, finally, in some language of logic. This study is concerned with the forms

directly associated with language and with the ways these forms are entrained and harmonized together through the processes of language. The uses of these forms, these structures and processes, for particular ends is a different study altogether.

Perhaps I should indicate very briefly why I think it appropriate that a study supported by the Department of Defense be oriented in this way. The defense establishment is organized, administered, and its obligations executed by human beings. And military, diplomatic and intelligence activities involve complex systems of interacting human personalities, whether in a neutral exchange of information, in a competitive situation or in an adversary relationship. We surely improve our position when we deepen our understanding of the instruments that humans are constrained to use when they think, speak and act. Secondly, an open-minded search for these natural instruments holds out the elusive but nevertheless real possibility that we may come upon fundamentally new technologies - perhaps technologies that have been used for hundreds of millions of years by organisms evolving in a not altogether friendly environment.

I will offer here a framework for thinking about the human personality and a meager hint about novel technology. Neither of these offerings is original in any important sense, but their integration may be of interest and may eventually yield technological benefit. Unfortunately I can offer little in the way of substantiation or justification, either theoretical or technological. The framework I will describe is the result, or outcome, of this study, and this report would be very different in substance had I had this framework available to me three years ago.

A study that is concerned with the relation between natural language and computation is one that involves a number of different aspects of the human personality. If it is to be nontrivial, it will necessarily be cross-disciplined, heterogeneous and eclectic. I will not offer now any specification for what I mean by "the human personality" since that notion is, in a sense, the subject of the entire study. I do mean it to include a variety of functions, processes and structures that are characteristically human if not uniquely human. This study is then a speculative pre-scientific contribution, rather than one which adds to scientific methodology or technology. But at the end I will propose a certain instrument which may have not only technological applications but may also constribute to empirical and methodological aspects of the study of human nature, of human language, and more generally of the role of computation in human activities. Beyond the possible experimental applications it is suggested that this technology may be useful in the development of military and industrial command and control systems, in man-machine interfaces, in computer-assisted instruction, and even in such a routine setting as a simple time-shared computer system supporting conventional scientific and business computations. In general, candidate applications may be found wherever human communication is involved.

2. Sketch of personality structure

I will presuppose a certain characterization of the human personality, one which I think is fairly unexceptionable, although its relevance to other issues may be called into question. It provides a framework which motivates the emphasis of this study and provides organization at certain crucial stages. I will suppose that each individual human personality is distinct and unique in important respects, but that there are formal or structural features that are common to every human personality. Individual variations are thus additions to, elaborations of and constrained deviations from this common norm. The generalizations I propose are not to be construed as assertions made with any certainty, nor are they necessarily empirically testable, although future insight may make some of them so. Tacit throughout will be suitable qualifiers such as "I think that ...", "It may be the case that ..." or "It is proposed that ..." to cite some typical hedges. I will speak of the idealized common part of the human personality as "the human norm", either to refer to an individual instance or in a generic sense.

The human norm has access, by orderly means, to an abstract, organized and stable record of its experience. I will not at the moment consider how this record is obtained from experience, exactly how it is organized, or how new experiences in succession are integrated into it. I require this record to be abstract since a consideration of the concrete record of experience belongs properly to the biological sciences and more particularly to neurophysiology in all its branches. In any case the issue is moot since at the present time we do not know how perceptions are related to physiologically defined sensations, or how the abstract organization of perceptions is related to these sensations and the physical structure of the organism. When neurophysiology provides us with a reliable and verified theory for the physical basis for memory, we will then have to relate the structures and processes of that theory to whatever we will then know about the stable, organized, abstract record of the experience of the human norm, how that record is used and what it is used for.

The human norm routinely and sometimes voluntarily engages in purposeful activity that in part determines a selection process acting upon the record of experience (the "RE"). One may think that voluntary purposeful activity can be accounted for solely with reference to behavior, but this thought ignores too many issues. Until we can account with some certainty for the relation between sensation and perception and between perception and motor activity physiologically described, we have no adequate or coherent link between voluntarily adopted purpose and behavior. This is simply because many of our thoughts and actions are organized with respect to what is in perception and very rarely with what is in sensation. It is likely that several distinguishable aspects of the human norm participate in an orderly way in determining this selection process. The process itself is a function in part of both the form and the content of RE. Additionally it may be conditioned by the behavior of the human norm,

and by anatomical and physiological characteristics unique to the particular physical eexpression of that norm. And it will certainly be conditioned by the environment in which the human norm functions, as well as by the processes that organize RE and incorporate new experiences into it. I will assume that conscious awareness is also essentially implicated with this selection process. In particular I will assume that we can be precisely aware of just one item from immediate experience or RE at a time, whether that item is simple (atomic) or complex, and I therefore also assume that the distinction between atomic and composite entities in RE can be made sufficiently precise. That is, the human norm has the capacity to particularize and localize its focus of attention on differentiated items in RE.

The human norm has the capacity to organize freely, and to reorganize, transient occurrences of fragments of its RE with reference to an orderly system of interacting preferential judgments. In a loose and informal way we may understand these to be described with reference to a number of dimensions, many of which are similar in character to what have been called "emotions". A direction is associated with each such emotional dimension, and it is this direction that gives rise to the concept of preference. Specific patterns of preference will not be part of the human norm. Each instance of the human norm will have characteristic preference patterns that vary with experience, past and present, and these patterns are also influenced by other factors in the human norm and its environment. But the capacity for patterns of preference will be universal - an essential aspect of the human norm.

Along the dimension of acquisiti~eness, for example, we will have the biologically normal disposition to acquire what is needed for biological survival, and beyond that we will have different degrees of desire, with extremes of greed, and beyond to obsession. Moving in the other direction we will have increasing indifference to acquisition, and in extreme cases a nearly complete neutrality or nearly total lack of concern. But the presence of the dimension indicates some degree of acquisitiveness, however slight. To eliminate all acquisitiveness is to eliminate the dimension itself. We also note that with respect to acquisitiveness, a human personality with total lack of concern is one with no felt need to solve problems connected with acquisition.

Along the dimension of aggression, for another example, we will have the biologically normal disposition to fight for survival when it is appropriate. Moving farther out on this dimension we find increasing disposition to more and more arbitrary argumentative attitudes and orientation toward violence, both physical and mental. But interestingly, as we neutralize aggression, we find, as before, increasing indifference and lack of concern. Along the dimension of submissiveness we have the normal disposition to cooperation, and in its extremes, the feelings associated with obeisance, fawning and self-humiliation. Along the dimension of cheerfulness we have a varying normal disposi-

tion to good humor, and in its extreme the disposition to believe unrealistically in the "best of all possible worlds". There are dimensions of comedy, tragedy, hate, love, generosity, vengefulness, guilt, innocence, temperance, and a vast host of others.

It is important to note that these dimensions do not constitute contrasting pairs of dispositions on one dimension. Greed and generosity are not opposite ends of the same scale. Indeed, we may find both factors operating simultaneously and selectively in the same human norm. Note also that I am not speaking here of behavior, of acts of greed or generosity, but only of a preferential attachment applied to selected dimensions, placing the human norm at least temporarily at specific degrees of preference. At the low end of each of these dimensions we find increasing lack of involvement with the emotion, lack of attachment. And finally we note that we may try to define composites - dimensions which are expressed as combinations (linear?) of others. And then preferential judgments may come to be associated with these composites. For example, we may express guilt as the sum of degrees of shame and responsibility, satisfaction as the sum of desire and gratification, and pity as a sum of aversiveness and compassion.

These dispositions to preference are universal in their form, although individual human personalities may make varying use of the general capacity. Other dimensions of human experience that admit of preferential judgment include believing, hoping, wishing, doubting, denying, suspecting, imagining, assuming, guessing, dreading, fearing, and on and on. The absence of each of these is in the specified respect a neutral state. In each case, as one moves further and further to the low end of the scale one encounters not complementary judgments but rather the absence of the judgmental attitude. The less one is involved with these scales the more objective can be one's perception of reality, and one's purposeful activities. And then, perhaps, the more detached one is found to be.

To say that the human norm refers its activities to preferential judgments is to say that it has the capacity to try to position itself at arbitrary points on these scales. It may attempt to increase its involvement with one (attachment) or to decrease its involvement with another (aversion). I will take these capacities for attachment and aversion as universal properties of the human norm. These two contrasting universals are then the origin of the directionality inherent in emotional dimensions.

There is also in the human norm a capacity to conceptualize or to abstract. It is an abstraction to distinguish visual from non-visual experiences. Red is an abstraction, as is squareness, triangularity and circularity, and straight lines and even points are abstractions. Softness, roughness and smoothness are tactile abstractions, and the notion "tactile" is itself an abstraction. In the auditory system, harmonies and dissonances are abstractions as are the notions of harmony and dissonance themselves.

The capacity for abstraction is so widespread, so inevitable, that we suppose a separate organ in the human norm to account for this pervasive and indeed decisive activity. We note that in particular this capacity to abstract may operate not only on direct sensations, but also on the dimensions of preferential judgments and their composites, as well as upon the results of the action of the abstraction agency itself.

There are, then, four capacities that are universal to the human norm - the capacity to select purposefully from the record of experience, the capacity to form varying degrees of attachment and of aversiveness, and the capacity arbitrarily to form abstractions and conceptual systems.

It is theoretically inherent in the capacity for abstraction that it has no apparent limit. The organization of experience into visual, auditory, tactile, olfactory ad gustatory categories is only the beginning. Even if we believe with Langer that the sense organs provide us with organized forms, we still recognize that occurrences of these forms are subjected to unbounded levels of abstraction. This abstraction capacity is at the foundation of science, art, ritual, religion, politics and all social institutions. At the personal level it is implicated with the control of waking life, with dreams, with structured institutions and with the role of the individual in them, with metaphor, with social deviance, with neurotic attitudes and with the variety of what we call "psychopathologies". It is the selective interaction of particular abstractions with the universal capacity for preferential judgments that produces the dimensions of the emotions. This is simply to say that preferences are not completely arbitrary. Their variation is systematically related to other aspects of the human norm.

3. The nature of meaning

It is a system of abstractions and preferential judgments, along with the current experience and the operation of the agency that integrates that experience into RE that determines at least in part the fragment of RE that will occur in conscious awareness, or that will constitute conscious awareness at particular moments, and these are also the factors that will determine the meaning of any particular utterance in language made by the human norm. In summary, these aspects of meaning are,

(i) a fragment of the record of experience;

(ii) a conceptual organization of this fragment;

(iii) a system of preferential judgments;

(iv) current experience;

(v) purpose - a primary preferential judgment.

In addition there will be the processes that integrate these aspects of experience, and the overall process of coordination among these processes themselves. These include

(*i) incorporation of current sense experience into RE;

(*ii) selection of particular preferential judgments;

(*iii) applications of selected preferential judgments to contents of awareness and its supporting unconscious or background substrate;

(*iv) organization of elementary units, at each level, into composites meeting conditions of relevance where relevance is related to systems of abstraction and systems of preferential judgments as well as to the current flow of sense experience.

(*v) integrative processes to organize and synchronize these mental processes..

I will need a specific term to refer to this arrangement of structures and processes in the human norm at any given moment and I will use the word "estate". We have a function, esteate which takes a moment of time as an argument and returns the norm's estate at that moment as its value. One of the tasks of linguistics is to relate these structures and processes that constitute the estate to those that are implicated in the use of natural language.

If we try to use current views about the nature of language to communicate with a digital computer using natural language, the computer will have to be able to construct, according to algorithmic design, a pairing between expressions in the selected natural language and well-defined objects in a domain representing all or parts of the estate, a domain whose structure is distinct from the structure of linguistic domains. That is, there is no reason to expect that the material which is the subject matter of language discourse will have the same structure as do expressions in the language. The subject matter of language discourse is constructed in part by the capacity for abstraction, and since this is an arbitrary capacity, we may not assume in advance what will be the nature of the relation between the structure of reality and the structure of the product of this abstraction capacity. But if an adequate pairing is to be obtained, both the domain of language expressions and the domain of estates must each be expressible as computable structures, and the pairing process itself must be explicitly articulated as an algorithm. These considerations may lead us, incorrectly, I think, to imagine that a purely referential or denotational scheme can capture what is important and valuable in the use of natural language for communication with a computer.

For example, if Exp is a set of expressions, atomic and composite, and Ob a set of objects and processes which occur in a natural estate, which expressions are to refer to, then with reference to a proper naming function Prop, we can define a reference function, ref:Exp -> Ob, in very general terms as follows:

> If x is atomic then (ref x) is (Prop x), and if x is the composite (x1 x2 ... xN), then (ref x) is
> (comp (struc x) (ref x1) ... (ref xN)),

where struc is similar to quotation in that it displays the syntactic structure of the unevaluated expression supplied to it. struc might be expressed, in slightly expanded form, more suggestively as a composition,

(struc x) = (syntactic-form (quote x))

comp is an extraordinary and interesting function. comp specifies how the reference of a composite expression x is determined not only by the references of its immediate constituents but also by the "grammatical" composition rules according to which x is formed out of its immediate constituents.

But we may raise questions about this general plan. What constitutes an adequate set of language structures for these purposes? What is the nature of the function comp? Is verbal text in fact the best means for communicating? What other kinds of signals are regularly involved in a communication event? Is communication with language facilitated by simultaneous reception of signals in non-verbal modalities? If so, how can knowledge of these be brought to bear on human communication? The problem with the ref function is that it tells us a great deal about how forms are manipulated if we already have some idea that what the forms are associated with can be "applied" to certain other forms. That is, the systematic form of reference and denotation presupposes an "applicative" language. But any association of this form manipulation with the substance of what the forms are associated with is quite arbitrary. It is true that the application of arithmetic functions to numerals can be uniformly described with a function like ref, but then there are no arithmetic functions in reality, and there aren't really any numerals either. These latter objects exist only as a consequence of our capacity for abstraction, and a certain social fluency for giving selected abstractions wide conventional currency.

Another problem with ref is its focus on immediate constituency. The more general notion of constituency is surely a valid one, but it is not at all clear how one would express a revised version of ref to account for the situation in which arbitrary constituents of an expression determine its value under ref. And perhaps more fundamentally, we might very well want a version of ref that allows the value of (ref x) to affect the values that ref will determine for constituents of x.

If the sets Exp and Ob are well-defined sets, then the function ref given above is simply a realization of what is called "model theoretic semantics" for a formal language. [Jardine, 1975] has shown that the plan to apply model theoretic semantics to natural language may make unreasonable demands on natural language. I think his point is that the truth condition for an arbitrary sentence in natural language may depend upon the truth or falsehood of other sentences in the language, and this is theoretically unacceptable for a simple reason. We lack a decision procedure for sentences in complete logical languages that include quantifiers. Other linguists and philosphers have seen other aspects of the attempt to use model theoretic semantics. [Potts, 1975] has observed that very little is added to the notion of meaning in natural language by model theory, because the system to which the function ref maps expressions must itself be just as well-defined as the system of well-formed expressions in the language itself.

In short, the question of meaning just gets shifted from one system to another without any substantial clarification.

Another issue related to model theoretic semantics and logic is that of "presuppositions". Let me use asterisks to indicate stress, one for minor stress and two for major stress. Then the sentence

John called *Mary a **virgin and then *she insulted **him.

is said to presuppose that to call someone a virgin is to insult that person. I think this is obviously wrong. In the sample sentence, what is infact communicated is the suggestion that to call someone a virgin is to insult her. It is part of the message itself, and is relative to the occasion of utterance. Other confusions surround the notion of presupposition. [Keenan, 1975] for example says that

> A sentence S logically presupposes a sentence S' just in case just in case S logically implies S' and the negation of S, ~S, also logically implies S'. In other words, the truth of S' is a necessary condition on the truth or falsity of S.

This is clearly inadequate, because the conjunction of the two implications is truth functionally equivalent to S'. The given condition for presupposition is thus simply equivalent to S' itself, and is therefore no condition at all.

One can say that the issue is not simply truth functional, but then we face the same problem that Jardine pointed out - to determine presupposition we have to have, in general, a procedure to decide consequence among sentences with quantifiers. I think that much of the notion of presupposition evaporates if we realize that a sentence ought to mean rather all of what it says. "Mary loves the puppy she found" doesn't preuppose that Mary found a puppy - it tells us that she found one. "Fred ate an-

other turnip" doesn't presuppose that Fred ate at least one turnip - it tells us so. I suggest that in the absence of strong and clear evidence to the contrary, we have to regard presupposition, as exhibited in these and similar examples, as a straw man.

And even more broadly we ask in what ways does a specification of syntactic structures restrict the facility with which we can integrate communication signals in a variety of different modalities. And indeed, what do we mean by "different modalities"?

In very broad and general terms we may ask how subject matter domains are to be represented in computational form. This question has received considerable and sophisticated attention not only in the field of modern linguistics but also in computational linguistics and artificial intelligence. To what extent are the schemes devised so far in those fields adequate? Given well-defined domains of language and of subject matter, what constitutes an adequate pairing? Is the denotational aspect of language the appropriate basis for the pairing algorithm? What are the alternatives? Are less well understood associations important, or even essential? If so, how best is the structure of an association net, for example, integrated effectively with the structures of language? Does this integration require specific new processes, or are cognitive processes already available for other purposes adequate? Is something excluded by adopting a denotational scheme? Does the use of a computer in principle exclude any important features of language and of communication, or more simply, expression in general?

The central idea in this discussion is of course that of adequacy. When is a definition of language stucture adequate? When is a definition of subject domains, or of a particular subject domain, adequate? When is a pairing algorithm adequate? And it is hardly surprising that these questions are so complex and so difficult to answer. The human mind is implicated in the domain of language and the rest of reality is implicated in the other domain. And the connection between the two involves the most basic and universal issues of thought and action.

It seems that there are two ways of arriving at a workable notion of adequacy. One may be described as a task-oriented approach and the other as an experimental approach. The task-oriented approach proceeds from a point of view that I think is fairly paraphrased as follows.

> The important thing about language is language in use, language to achieve an effect or a desired result. If a computer system for language "works", if it does what we want, then that success is itself important confirmation that we understand at least as much language as is involved in achieving that success. To understand is to know how to use.

An apparent advantage of the task-oriented approach is that it seems to have an operational character, and operational ideas are always in favor.

On the other hand, we may adopt a more strict experimental approach. The goal here is to understand the structure of natural language as it really is, the structure of reality, and the nature of the connection between them (presupposing that there is such a connection). This will predictably be a difficult issue. Language and mind are so intimately involved that there seems to be no objective, reproducible set of experiments that can inform us about those aspects of language that are essentially related to mind. Distributional linguistics was conservative in this regard. Its studies, some decades ago, consisted largely in organizing linguistic data, with careful attention to the exclusion of subjective judgment and mental attitudes. At least, the desirability of this exclusion was kept in the foreground. Chomsky's advance over the traditional theory was to propose that linguistic phenomena manifest, in an observable and predictable way, structure that leads us to conclude first, that language is administered by rule by some kind of definite, "effective" process, and second, that from the manifest details of this structure we can properly infer something non-trivial not only about the effective processes and the agent that administers them, but also about the mind that is involved with them.

It is important to appreciate that modern linguistics is as much a hard science as is any "physical" science. The apparent phenomena of position and momentum dominate physics, for example, along with the slightly less apparent phenomena of radiation and electric and magnetic fields. Apparent speech signals dominate language, along with the slightly less apparent phenomena of cognition, feelings, attitudes, intentions, purposes and thoughts. Physical science has its common basis in the entirely fictitious creations called numbers, equations and deductions. There used to be, in France, a standard meter stick. But there never was, in France or anywhere else, a bottle with the standard zero in it. There are no numbers anywhere in physical reality, just as there are no equations and no deductions. These are "useful" abstractions, helpful in organizing our description of that part of reality that is characterized in terms of position, momentum and other similar phenomena. The bits and pieces of our perceptions of these phenomena are organized by arithmetic, mathematics and logic. In an exactly similar way, the bits and pieces of linguistic signals are organized into categories and relationships that are theoretically constructed and then experimentally confirmed or disconfirmed by further direct observation, or by inference from direct observation. The bits and pieces the linguist starts with are physical observables (the measurables corresponding to mass, charge, position, momentum, etc.) and his arithmetic is a greatly extended but still quite precise system of symbol manipulations (as, of course, is the case with the application of mathematics to the physical sciences). Modern generative linguistics proposes several levels of structure, and it is a fact that the more remote a structural

level is from the raw phonetic data, the more difficult it becomes to get agreement about verification of prediction. At abstract levels of linguistic organization, one occasionally comes to depend on an implicit feeling of recognition, a subjective response, in order to accept the relevance of an experimental observation.

Here, for example, are contrasting pairs of utterances which are said to induce recognition of systematic variation in form and interpretation.

They are harvesting wheat.
They are dazzling insights.

Revolutionary new ideas appear infrequently.
Colorless green ideas sleep furiously.

To prove that theorem was difficult.
Proving that theorem was difficult.

Those men are visiting dons. Visiting dons can be boring.
Those folks are groveling dupes. Groveling dupes can be boring.

Last night I suspected that Tabby ate the mouse.
I suspected that Tabby ate the mouse last night.

This picture was painted by a real artist.
This picture was painted by a new technique.

The escaped prisoner was caught by the gendarme.
The escaped prisoner was caught by ten o'clock.
The escaped prisoner was caught by the student union.

I found the man eating in the cafeteria.
I know the man eating in the cafeteria.

The police were ordered to stop loitering after midnight.
The police were ordered to stop patrolling after midnight.

One can see that, prejudice and habit aside, the situation is no different in principle for the physical sciences. This is because what we choose to describe with numbers, equations and deductions is arbitrarily selected.

It is of course a completely open question whether or not there are mental structures that are not describable with reference to radiation, say, or position and momentum. And if there are we would hardly be surprised that we cannot detect them with the instruments of physical science. In principle, the correct way to determine if there are extra-physical phenomena is to formulate a neutral framework in which neither conclusion is assumed, and in which external events can effect a determination. I do not know of such a framework, but we may keep an open mind into the future. The issue does have consequences. Fodor writes for example that

> ...the operation of a sensory mechanism in responding to a physical property of an environmental event is an empirically necessary condition for the organism's perception of <u>any</u> property of that environmental event.

This, as an assumption, is surely incorrect, for physically determined responses can account only for physical properties, and these only relative to a conceptual scheme provided by the capacity for abstraction. And in the long run we may have to extend the scope of "sensory mechanism" so far as to make the notion almost meaningless.

There is a third approach to the study of language, an approach which has received little or no attention to date, although it may, in the long run, have considerable impact on our understanding of human communication in general and in particular on those communication systems that involve humans and computers. In general terms it proposes that language may best be understood as a process whereby mind and physical reality are integrated together to meet some universal condition of coherence or consistency. We can better appreciate the point of this proposal when we have reviewed in more detail the serious studies of language that have taken place in the last quarter of a century, both in the task-oriented approach and in modern generative transformational grammar. We will then be able to ask whether or not this approach can be instrumented in a concrete way.

There are two deficiencies with the task-oriented approach. One concerns the choice of a decision-making agency to decide which tasks should be pursued. In principle, the "it works" criterion is completely arbitrary. We find it socially unacceptable to allow a psychopath to exploit this principle in his daily life. The question is, who is to decide what it is that is to work. (We say that in a democratic society, short run difficulties aside, we can count on the voice of the majority in the long run to make proper choices. But science is not and should not be a democratic affair. Science is socially useful precisely because it is neutral, objective and experimentally verifiable.) But a deeper flaw with the task-oriented approach is that it may not ask questions that are experimentally confirmable. For example, a sleight-of-hand artist may be observed to pick a lit cigarette out of someone's ear. But we know on other, quite reasonable and reliable grounds, that cigarettes cannot literally be picked out of anyone's ear. What we conclude is that our observation in this case was in some way deficient. In an exactly similar way, a computer system may give the illusion of understanding some kind of natural language, probably by virtue of its behavior, taking the notion of behavior in a loose sense. But our observation of this behavior, and our interpretation of it, may not be held relevant to the understanding of naturally occurring language in the absence of some kind of objective, experimentally confirmed and theoretically based treatment that relates natural language, the biological instrument of illusion, to the computer, the artifactual instrument of illusion. This re-

quirement and some condition of coherence are the most basic requirements of science. And it hardly needs to be said that the task-oriented approach cannot be justified as anything more than technology simply by a majority vote among a population of specialists. We simply have to to distinguish between types and degrees of illusion and of reality.

If the final goal is to produce technological artifacts that adequately execute a well-defined function, then we can say, in a number of cases, that this goal is met. But if the goal is to understand the role of computation, algorithms and computable structures in naturally occurring phenomena, then computer technology has contributed almost nothing to that goal. This question is of some importance and in the next section we will discuss it in some detail.

4. Computing as a process

In order to confirm that computation is an essential part of a natural phenomenon we would have to make objective, repeatable observations verifying the following criterion:

> Criterion I. A structured physical object is a computer if we can identify among its constituent parts specific substructures which interact to produce recognizable computations.

We can state this in a more specific way.

> A structured physical object is a computer if we can describe its structure and function in such a way that it is seen necessarily to administer the manipulation of forms, where the forms and the patterns of manipulation meet theoretically specified conditions of definiteness, finitude, stability, and so on.

We can sketch very briefly here the kinds of issues that are involved in such conditions. There are conditions on the elementary objects, or atoms, that enter into computations, and on the composite objects that have those atoms as ultimate constituents. There are primitive acts, and composite acts. In one form or another there will be a "fetch-execute" cycle, a systematic means of administering control structures - of managing composite acts. There will be input and output mechanisms and there will be memory. I will treat just a few of these here. The atomic objects that enter into a computation are its irreducibles or atoms, and it is with occurrences of these that a computer deals. The computer may recognize no internal structure in these occurrences of irreducibles. The computer must be able to inspect a candidate (occurrence of an) irreducible, to identify it, to create a dstinct occurrence of it, and even to annihilate occurrences. An occurrence of an irreducible must be

bounded in extent, on all physical dimensions. Its position and momentum must be exactly known, or at least must be known to within bounds that are set in advance, and its position and momentum must never stray outside those bounds. If the computation proceeds in space and time, then computational characteristics may not depend essentially upon spatial or temporal factors. If computation refers essentially to energetic factors, then it must be independent of any particular quantitative energetic measures. All relevant physical configurations must remain invariant throughout the requisite lifetime of the computation.

Composite objects must exhibit an immediate constituency relation. Any composite must have only a finite number of immediate constituents. Composite objects must be semi-grounded - any progression through immediate constituents must terminate after encountering only a finite number of occurrences of distinct irreducibles. That is, the total number of distinct immediate constituency relations must be finite in number. But there may in principle be no physical bound on this number. The class of composites must be potentially infinite. We note that this condition is not in general expressible as a computable condition. Strictly computational steps cannot determine in a finite way of a physical structure whether or not it is of infinite extent.

The primitive acts that a computer can commit fall into two classes - judgments and operations. The computer must have a means for directing its attention to specified occurrences of objects, and it must be able to tell, of an arbitrary occurrence, whether or not it is an occurrence of an irreducible, and if it is, which one occurs. It must be able to generate on demand a fresh occurrence of a specified irreducible in a time and place distinct from those of the original occurrence of the specified irreducible. (We note that if there are naturally occurring computations, then this last requirement implies an organizing principle in reality that strictly transcends space and time.) Each primitive act must require fixed resources to be initiated, carried through and concluded. In general, the computer must be able to behave in a synchronous or asynchronous manner on demand. Upon conclusion of a primitive act the computer must be able to return its attention to where it was at the initiation of that act. And it must be possible to make this return contingent upon the identification of a specified occurrence of an irreducible in a specified configuration. The computer must have access to a memory whose structure remains completely fixed for the duration of a computation, except as freely modified by the computation itself.

Every general purpose computer must be able to execute any of an infinite class of distinct algorithms, and every algorithm must be expressible, or representable as a computable structure. And for every general purpose computer there must be a universal algorithm for that machine - an algorithm which accepts as inputs representations of algorithms and inputs to algorithms, and which interprets in a step by step way exactly those acts which the

computer itself is capable of committing.

From these considerations we can easily appreciate that we determine of something that it is or is not a computer not by examining and interpreting what it does, but by inspecting its physical, temporal and energetic characteristics. And we determine of a structure that it may or may not represent an algorithm by determining that it is in fact implicated in a precisely defined physically observable way with the activity of a physical computer. These are the kinds of considerations we face when we ask of something whether it is a computer, an algorithm, a computational structure, or a computation.

It is clear that all grammar rules proposed so far within the modern tradition of generative grammar are computable rules, and it is equally clear that all the proposed conditions on control structures for administering the application of these rules can be met by suitable algorithms. Insofar as linguistic data confirm the reality of these computable entities they also confirm the existence of computational characteristics either in the mind or in the brain, or perhaps in both. We cannot investigate this matter in any depth here, except to make two observations. One concerns the nature of the physical universe and the other the nature of nervous systems. Those who are familiar, even at a superficial level, with quantum physics will appreciate that it is far from obvious that there can be any computers organized out of elementary particles. And one cannot help but notice that the organizing principles that seem to operate to produce organic systems are in many respects similar to those that characterize a computer. It may be the case that computers can only be organized at the classical level. But then, of course, we are entitled to ask why or how the universe is organized so that indeterminate atomic processes when massed in sufficient aggregates assemble themselves into agents that compute.

5. Mind, brain and computing

Direct observation to determine whether or not nervous systems meet computational conditions have not been made, as far as I know. Experimental observation of this kind does not fall in the province of either linguistics or of artificial intelligence and computational linguistics. More generally, I know of no objective experimental confirmation that there are any identifiable biological computers, even such as might be implicated in the structures and processes of language. Analogies between brain and computer are so loose and superficial as to be scientifically meaningless. It is true that organized neural nets can be systematically related to behavior, but it is also true that the function of these nets is affected in essential ways by electrical, chemical and even possibly mechanical processes that are continuously graded. These are simply not digital events. The reality appears to be that the brain is so intimate a blend of digital and non-digital processes that important and even fundamental issues may be ignored by focusing exclusively on the digitally computed aspect, if there is one. Much is made of the

all-or-none character of the neural firing event, but there are distinguished neurophysiologists who question its importance.

> ... the role of impulses in the central nervous system in representing and transforming information has seldom been established and is nowhere investigated to a satisfactory degree of completeness. Moreover, several kinds of evidence, although somewhat indirect, point strongly to the importance of other, nonimpulse vehicles for carrying information in the brain according to their corresponding coding schemes; the importance of such nonimpulse codes may well surpass that of the 'classical' nerve impulse. ... The suggestion has been made and must be entertained seriously that it is the impulses that are best regarded as the epiphenomena (at least in some parts of the central nervous system) and that only through understanding the properties and interactions of the electrical waves with the anatomical substrate will we arrive at a satisfactory understanding of the higher behavioral and mental processes." [Perkel, 1969]

And speaking from the point of view of a specialist in sensory research, Ragnor Granit writes "Whatever the nature of the central mechanisms, they must be capable of interpreting the frequency code". [Granit, 1955]

It might be suggested that digital computers can easily be prepared to compute approximations, to any degree of accuracy, of continuous structures, but if this were the case the problem of explanation would still be unsolved. We would then need to understand how and why organic tissue should evolve in such a way that it tries to construct digital approximations to continuous phenomena. And even in general terms , there are serious issues connected with the effort to locate a computing device in the brain. In the first place, organic tissue, and this is as true of brain tissue as of any other living material, is never the same from moment to moment. Even those structures which are directly involved with membrane currents are constantly changing, are undergoing a continuous process of breakdown and reconstruction. And in the second place, even at the level of abstract thoughts, the accumulation of experience induces a different continuous progression in both mental and biological structures. Whatever physical invariants we might find in the brain, they are almost certain to be found only at relatively macroscopic levels of neural integration. No matter how widespread the current view to the contrary, it remains a scientific fact that there is at present no direct experimental evidence that there is anything even resembling a digital computer at any significant level of organization in the brain.

I have devoted some space t. this issue in order to strengthen the suggestion that an investigation of natural or biological systems, to determine exactly how they compute whatever it is they compute can only be of value to us in understanding how to make the most out of computing instruments. When compared with human technology, biological systems have been enormously successful in representing their environments and in responding to them in meaningful ways.

6. Computability in Grammar

Generative linguistics provides a scientific and experimental point of view for the study of language, and derivatively for the study of the mind. Observable speech signals and manifest regularities in those signals provide the data for linguistic studies. These regularities are easily detected in the various speech signals produced by an individual, and also in the speech signals produced by fairly large classes of individuals. At the very least, generative linguistics provides a framework for the organization of these data. And to the extent that the unprejudiced observer finds this organization compelling, he will be prepared to accept the assumptions about natural language that constitute this framework.

One very basic assumption is that the linguistic capacity is intrinsically organized for computation, in important respects if not in all respects. Grammar rules are typically written as Post productions, and such constraints as are placed on the order of application of such rules are clearly computable. As a consequence of this assumption we have the further assumption that in the individual, this organization can be expressed at an idealized and abstract level by a finite set of rules (Post productions) which specify in principle a certain competence in language. An agent is competent at a language if that agent is able to distinguish by appeal only to those rules, those "forms" that are admitted by the rules from those that are not. [Chomsky, 1957, 1965, 1968]

In broad but accurate terms, then, competence in language is expressed by two things - a finite set of rules, and a mechanism for the administration of these rules. In generative linguistics, one states these rules as Post productions, as operations on symbol strings. These operations must include primitives to recognize whether or not two substrings are the same, to delete a substring and to insert a substring. And they must include a means to fix attention, arbitrarily, on specific strings and substrings, in order to make possible the stable realization of the operations. One might think this a restriction. There are, after all, several other formulations of the theory of computation. We have Church's lambda calculus [Church, 1941], Turing machines [Turing, 1937], the equation calculus [Kleene, 1936], Schoenfinkel's system of combinators [Schoenfinkel, 1924], and others. And foremost among them all is the logic of truth-functions, predicates and quantifiers - first order logic. Well-known fundamental theorems establish that each of these

theories is equivalent to each of the others in well-defined and important senses. But Post productions [Post, 1943] and Turing machines have a unique status. Neither of them is defined to accommodate any abstract system of objects and operations. Church's lambda calculus has as models domains of functions and the notion of functional application. The equation calculus is concerned with inductive definition and computation. The "feel" of a Turing machine is that it commits physical acts upon a physical environment. (It makes marks on a tape, erases such marks, and moves its read/write head back and forth across its tape.) In all other formulations the fundamental operation is that of substitution - to replace a specified occurrence of an expression with an occurrence of some other specified expression. Post productions realize this substitution on symbol strings, while in the other formulations the operation is more abstract, and is supposed to apply to syntactically specified complete expressions. Post productions are minimal in the sense that they make no assumptions about the uses to which symbol strings are put, and they do not assume that the operations are to be realized as physical acts, as does a Turing machine.

With respect to the computing agent that underlies language, generative linguistics says nothing at all except that ordering of rule application seems to be a reality. Certain rules may only be applied once, certain of them must be applied before others, and certain sets of them may be applied repeatedly, in order, in a cyclic way.

Here is a synoptic description, such as a generative linguist might propose, to "account for the organization he finds in linguistic data. The competence that a normal human being manifests in his use of his native language is idealized as a set of interacting components. These communicate with each other by generating and manipulating certain well-defined structures. In every case both the structures and the patterns of manipulation must meet constraints that identify them as computational structures and computational processes. There is a "lexical" component that accounts for the properties of words and word stems and for some of the relations and associations that appear to hold among words. It also incorporates information that accounts for observed occurrences of these words in linguistic structures. For example, "fellow" is marked in the lexicon as a noun, as an animate noun, a count noun, and as indicating the feature "human". These markings exclude from the class of grammatical sentences such sentences as

We should fellow that fence.
That wasn't much fellow.
The fellow excused herself and left.
That fellow took off its hat.

Further, for the sentence "This fellow believes himself to be a genius." "this" is marked as a determiner, and as singular. "Genius" is marked as a noun, as singular, as animate and as conscious. "Believes" is marked as a verb, as in the third per-

son, as singular, as present tense, as taking a conscious subject, and as having a complete assertion as object. But there are other constituents for a lexical entry. A distinguisher incorporates the idiosyncratic material for a lexical entry, that material which will have no effect on grammatical processes. For example, the lexical entry for "ball" will include the category "Noun", the grammatical feature that it is a concrete noun, the semantic feature that it is a physical object, and the distinguisher that it is used in sports, entertainment, and so on. Once these lexical items have been inserted at the terminal nodes of a structural description provided by the categorial component, a set of projection rules is invoked to associate amalgamations of the semantic information found in the lexical entries with the non-terminal nodes of the tree representing the structural description. One infers that there would be a distinct projection rule to accompany each production rule in the categorial component. Chomsky later organized grammatical features into "complex symbols" which conditioned the possibility of lexical insertion in structural descriptions, a conditioning that took into account other lexical insertions into the same structural description.

In the "base" component there is a phrase-structure component that represents the form of elementary, primitive or minimal assertions.

The man has a dog.
We are fast.
Yumyum loves Wally.
She lost patience.

There is a transformational component that accounts for some of the complexity of actually occurring expressions and for the relationships that obviously hold among certain sets of different expressions. All the following, for example, are closely related.

That fellow was computing sums.
Sums were being computed by that fellow.
Who was computing sums?
What was that fellow computing?
What was that fellow doing?
That fellow's computing sums ...
The computing of sums by that fellow ...

And there is a morphophonemic or phonological component that accounts for preparing an expression for utterance as a sound pattern.

One also specifies a "semantic component", say, one which associates in an orderly way a "reading" with each deep structure.

For example, so called "projection" rules will project a semantic "reading" from the lexical items "up" through more complex constituents of the expression. For "Eddie gave his little

doggie some yogurt." the "little" and "doggie" will be "combined" by a projection rule to give a reading for the phrase "little doggie", and "some" and "yogurt" will be combined to give a reading for "some yogurt". The particular combination rule used will be determined by the grammar rule that generated the phrase. From the readings for the phrases "Eddie", "gave", "his little doggie" and "some yogurt" the reading for the complete sentence will be obtained.

Five kinds of linguistic forms appear in this description. The phrase-structure component generates (computes) skeletal structures which specify the forms of simple assertions. The lexical component inserts lexical items into these skeletal structures at specified positions, providing what is called a "deep structure". The transformational component turns deep structures into "surface structures", which are then manipulated by the phonological component to produce nearly utterable matrices of sound specifications. And the semantic component associates a "reading" with each deep structure. Within this framework, all decisions about the forms of particular rules and about the order of rule applications are referred either to linguistic data, or to considerations of simplicity and economy of theoretical assumptions. Thus there is a phrase structure pattern, a deep structure, a surface structure, a phonological shape and a "semantic reading".

Consider, for example, the sentence "That fellow believes himself to be a genius." The categorial component will generate the following,

((Det Noun)(Verb ((Det Noun)(Verb (Det Noun)))))

from the rules

Sentence -> Nounphrase Verbphrase
Nounphrase -> Det Noun
Verbphrase -> Verb Nounphrase
Verbphrase -> Verb Sentence.

Lexical insertions will then produce the structure

((That fellow) (believe ((that fellow) (be (a genius))))),

and after a number of applications of grammatical transformations we have

((That fellow) (believes (himself (to be (a genius))))).

These components and the mechanism which integrates their handling of linguistic structures may be called the "language generator". According to the most basic assumption, this gener-

ator acts in certain essential respects like a computer. This observation has important consequences, but in order to state and appreciate those consequences we will have to consider further some fundamental aspects of computation.

7. Abstract Theory of computing

The centerpiece concept in computability theory is the notion of a computation. A computation is a finite sequence of states, each of which is related to its immediate predecessor in the sequence in a definite way by exactly one of a finite number of rules. A state is a definite and orderly arrangement of a finite number of occurrences of irreducibles. A theory of computation is made more explicit by giving, to a suitable degree of definiteness and exactness, clarifications of the concept of a rule (relating states) and of the concept of orderly arrangement, and perhaps of how the process of transition through a sequence of states is controlled. A definition of a particular computing system must specify a class of structures which count as states and constituents of states, and must specify in all essential respects what a rule is and exactly what state is produced as a result of applying a rule to any specified state.

From this point of view a computing theory is a pair [S, R], where S is a class of structures and R a finite set of rules. S is of course itself a complex system. It includes a family of predicates which can distinguish among distinct computer structures, a family of selectors which can pick those structures apart, and a family of constructors which can assemble new structures out of given ones. If an application of some rule in R turns a state s1 into a state s2, we indicate this fact by writing

R:s1 -> s2

If some finite number of applications of rules in R turns a state s1 into a state s2 then we write

R:s1 ->> s2

Some rule systems have the property that given any state, at most one successor state is determined by the rules. That is, at most one rule applies in exactly one way to any given state. Such a rule system is deterministic or monogenic. The contrasting terms are polygenic and non-deterministic. If a sequence s1, s2, ..., sK is a computation, we call s1 its initial state and sK its final state. If there is no state s0 distinct from sK such that R:sK -> s0, then sK is the ultimate state of the computation s1, s2, ..., sK. We note that the structures used to organize states need only meet very specific conditions of definiteness, recognizability and stability (at least throughout a particular computation). And therefore, rules need only meet the same kind of conditions of exactness - it must be obvious that the application of a rule always produces the same definite, predictable and recognizable result. These rule systems determine how the computer

acts no matter what program or what data structures it surveys. A particular pattern of rule applications may be specified externally. Such a pattern is called a program, and it must be possible to express any of an infinite class of programs to any particular general purpose computer. That is, among the computer's structures, some will be recognizable by the computer as patterns of rule applications.

In order to speak systematically about the use of particular computations we introduce two further constructions. There is a function geninitial which takes as arguments a pair of structures, one representing a program in the language for the computer system, and the other an input to that program. And we have a function outcome which, when applied to an ultimate state produces a structure occurring in some specified position in that ultimate state. geninitial is a "constructor" function, and outcome is a "selector" function.

Suppose now that f is a function of any kind, whose domain is A and whose range is B. We imagine that some functions can be computed and that some others cannot. Thus we need a definition that determines under what circumstances a function is computable, according to a particular computer system [S, R].

> The function f:A -> B is [S, R] computable if there is a program p, and a representation function rep:union(A, B) -> S, such that f(x) = y if and only if
>
> geninitial(p, rep(x)) ->> Y
>
> where Y is ultimate and outcome(Y) = rep(y).

We notice that without the function rep it is nearly imossible to speak of a function's being computable by a particular computer system. It may be possible to define the class of computable functions in some other way, but to define computable relative to a particular computer system seems to require a well-defined representation function.

With these background developments we are now in a position to make some observations.

If we have observed a physical process from its inception to its conclusion, we do not imagine that we have characterized the process solely by describing its outcome. If the outcome of such a process is the sum of two numbers, we may not then infer that a digital computer was responsible. Any general purpose computing agent can do arithmetic, but we have no a priori grounds for asserting tht only computers can do arithmetic. To tell of a process that it is a computation, or that it is in important respects a computation, we must first determine that in relevant respects that process can be analyzed into a finite sequence of discrete states. Then we must determine that these states are digitally structured in a systematic way. Then we must determine

that consecutive states in the process are related in a systematic (rule-governed) way. Then and only then may we conclude that it is an [S, R] computation, where S expresses the orderliness in the state structure, and R the orderliness in the state transitions.

But on the strength of these observations alone we are still not justified in concluding that any particular computable function has been involved. If we want to say that the process is a computation "of" some particular thing, a computation "of" a particular function, then considerably more is needed. First it will be necessary to recognize a certain kind of orderliness in the function which we believe has been computed. Then we will have to have a convention that such structure is represented in precisely specified computational forms. We will have to recognize that the initial state of the process could in principle have been constructed by some version of geninitial, and that outcome applied to the ultimate state would yield the result. Indeed, we would have to recognize the ultimate state itself as terminating the process. All these conventions are matters of interpretation, and contribute in no essential or even interesting way to the bare notion of a computation.

A complete characterization of computing specifies a well-defined set of forms, a well-defined set of programs, and a well-defined set of computations. From the point of view of computing alone, the forms are uninterpreted, and the programs have no meaning except as form manipulations. And a computation is entirely without meaning except as a record of the actual carrying out of the complex form manipulation specified in a particular program. In short, computation is the meaningless manipulation of uninterpreted forms.

A computation proceeeds according to specification whether or not this specification accords with any expectations that might be part of the interpretations we place on the structures and programs that determine that computation. There is no place in this picture for preferential systems or value judgments. We judge one computation to be good and another to be not good on the basis of our own interpretation of it, together with other subjective and social considerations. We cannot define, solely with reference to the manipulation of forms, such complementary notion pairs as good/bad, right/wrong, sad/happy, cruel/kind, elated/depressed, calm/agitated, belief/doubt, greed/generosity, ennui/interest, and so on. Form manipulation cannot itself be said to be humorous, nervous, hopeful, deluded, gentle, aggressive or shy, despite a certain human disposition to find pleasure in puns.

Even the concept of causality is not easily related to computation. Since computation can be realized by any of a variety of physical mechanisms, computation is independent of any particular aspect of physical reality. Therefore, the existence in nature of computations would imply the existence of some non-causal agency or principle which directs that uninterpreted form

manipulations unfold sequentially according to rule. The sole function of computation is the sequential, rule-governed unfolding of form manipulation.

Before we return to a discussion of grammatical and linguistic ideas we pause for one very fundamental observation. The idea that a class of phenomena manifests computational properties is only significant if there is an alternative. It must in principle be possible for matters to be otherwise. Our understanding of computing is deepened and enriched by the formal studies from the decade of the nineteen thirties, and by such studies as those that relate logic and computing more intimately [Robinson, 1979], [Kowalski, 1979], and those that attempt to describe computing in strictly abstract mathematical terms[Scott, 1970]. But these theoretical studies are important in part because they draw a distinction, a distinction between what is computable and what is not, and this distinction may be of considerable importance. Linguistics, artificial intelligence, and even broad areas of modern cognitive psychology rely essentially for theoretical constructs and for experimental methodology on the concepts of a computer, a program and a computation.

In these cases it is scientifically and intellectually inadmissible to ignore fundamental aspects of computational ideas. The assertion that "the line between a computer and a non-computer need not be sharply defined" represents nothing more than a decision not to ask certain questions, and to ignore discussion of certain issues. [Putnam, 1979] What would we think of a physicist who said that the line between states of motion and states of rest need not be sharply defined? A decision to remain uninformed about central concepts is unjustifiable on any basis.

If the concepts of computing are to be significant in explaining physical processes then there must be alternatives. Fields are physical systems which are not digital or computational in form. Waves such as are described by the Schroedinger equationsa are not digital in form. Much, in fact, of what physical science describes at the classical level is not in computable form. It is quite reasonable to ask whether computational structures and processes develop in some orderly way out of non-computable substrates, and if they do, what agencies administer this development. And what physical principles affect such development? And what is the relation between physically manifest causality and the sequential unfolding of a computation?

8. Interpretation in Grammar

We are now in a position to appraise the idea of interpretation in the theory of grammar. We have seen that a computational transaction is entirely void of interpretation or meaning except as the orderly manipulation of forms. If we choose to place any interpretation, any meaning, in a systematic way, on each of a specified class of computational objects, we may have to be prepared to make use of non-computational structures and

processes. Within the generative tradition in linguistics there have been two major proposals for relating grammatical structures to meanings. One has been called "interpretive semantics" and the other "generative semantics". We can grasp the supposed difference in the following way [Chomsky, 1971]. A generative grammar for a language specifies for each grammatical utterance in that language a finite sequence P1, P2, ..., PN of structures, which are more definitely called "phrase markers". Any such sequence must meet three conditions. (1) PN is a "surface structure", which means that only morphophonemic rules may be applied to it. It is, in a certain sense, an ultimate grammatical structure. (2) For each i, Pi and its immediate successor are related by one of a finite list of transformations, functions which map phrase markers to phrase markers. (3) There is no P0 such that P0, P1, ..., PN meet conditions (1) and (2). The interpretive semantics position is that P1 is generated by the categorial or phrase structure component of the grammar, and that for some I, P1, ..., PI are obtained solely by lexical transformations, rules that insert lexical items into the output of the categorial component. And then all phrase markers after PI are obtained solely be non-lexical, grammatical transformations. This PI is called the deep structure. Deep structures determine semantic representation, express grammatical relations, and reflect constraints on co-occurrence of lexical items. And the deep structure determines the surface structure by way of the system of grammatical transformations. The surface structure PN in turn determines the phonological shape of the utterance.

The generative semantics position is that there is no deep structure, and that there is no distinct semantic component which maps deep structures to meanings [Lakoff, 1971, 1972]. It holds that lexical and transformational processes are intermingled, and cannot in principle be separated out in the way described by the interpretive position. For this view, all the semantic information which the interpretive position provides can be as well supplied by allowing lexical and grammatical transformations to be freely applied, in systematic ways.

Another way of drawing the distinction is to say that for the interpretive view, the rules that associate a semantic "reading" to an expression are of a fundamentally different kind than are those rules that carry out lexical and grammatical processes, while the generative semantics view holds that only transformations are needed. Although the "projection" rules which the interpretive view provides are different in kind from the grammatical rules, they must still meet quite specific conditions of precision, definiteness and finiteness. If we understand "generative" to mean "computational" as I think we must, then the difference between interpretive and generative semantics is a difference in choice of computable structures and in the organization of the effects of different computing systems. On the other hand, if the projection rules envisioned by the interpretive view are understood to include structures and processes that may not meet the computational conditions of exactness, definiteness, invariance, and so on, then there is indeed a funda-

mental difference. Within modern linguistics I am aware of no proposals for non-computational structures and rules for the semantic component, but Chomsky has commented on the notion of deep structure and on the relation between syntax and semantics that

> No area of linguistic theory is more veiled in obscurity and confusion, and it may be that fundamentally new ideas and insights will be needed for substantial progress to be made in bringing some order to this domain [Chomsky, 1971].

One may try, of course, to determine whether linguistic data provide any evidence bearing on the question whether the semantic component of a grammar is realized in terms of computable structures and processes that have forms distinct from those associated with the other structures and processes of the grammar. The active debate between the interpretive and the generative proponents in fact reports these efforts. But this much we can say, whatever the outcome of that debate. Insofar as semantic factors are incorporated into a grammar, either by means of transformations, as the generative semantics view holds, or by means of projection rules of a different kind, as tbe interpretive view holds, any question of "interpretation" in its conventional mathematical sense is moot, because, as we have seen, computing is the meaningless manipulation of purely uninterpreted forms, and given the generativity assumption, the same can be said of any orderly system of phrase markers, semantic readings and phonological shapes. Insofar as meanings refer to goals, purposes, preferential judgments, attitudes, emotions and so on, some extra-grammatical non-computational agency must necessarily be involved.

9. Structure of mental images

In the light of this discussion we can make a justified generalization. The conceptual boundaries of a generative (computational) description of language delimit that aspect of human mental activity that is void of meaning, at least to the extent that meaning is related to interpretation in the usual sense in which a "formal system" receives an interpretation. And if one takes seriously the task-oriented studies in artificial intelligence, including "computational linguistics", the same may be said of them. Insofar as meaning is implicated with values, judgments, purposes and attitudes, the effect of an adequate generative (computational) description is to determine, at least in part, where we need not look for meaning.

Another question of interest may be put quite simply. Who writes these linguistic programs, these systems of rules? Who decides which programs to write? We note that this question is not answered by referring in some free form way to the biological notion of evolution or natural selection. Natural selection is not concerned with the manipulation of pure, uninterpreted form. It is quite value ridden. Survival, the enhancement of the ca-

pacity to regenerate, is a matter of preference, and we may ask why there is a preference. (Note that it doens't matter who or what "has" the preference.) If we say finally that things are simply the way they are, we are then only deciding not to organize the data of nature in any significant detail.

Other questions arise in connection with the notion that the mind deals with computable structures. One concerns the nature of mental imagery, another the nature of memory, and another very important issue concerns the way in which information is transmitted into the mind. The concepts of attention and awareness play an essential part in all these issues and it may be necessary to relate them in a systematic way to the mental computer if there is one. In addition we recognize that there are a variety of other human mental activities which make use of facilities that appear in language processes, facilities such as imagery, memory, articulation, and the distinction between deep and surface structure. And the whole range of psychopathologies that can and do affect mental life must be systematically related to the array of mental functions that constitute the human norm.

Consider, for example, the situation in which an individual perceives a set of objects organized into a situation. It is taken for granted that initial information processing stages involve the external sense organs which are characteristically stimulated by physical signals originating in the environment. These stimulations are then transmitted through afferent pathways into the central nervous system, and in the course of this transmission may undergo structural transformations about whose nature, extent and purposes we can presently do little more than speculate. We further imagine that somewhere and somehow in the higher brain centers the upshot of this transmission - transformation activity stabilizes in perception. For example, as a consequence of an elaborate system of expectations, complex neural events, and a rich substrate of visceral and chemical events, we perceive before us an aromatic, medium-rare roast of beef.

There are two difficulties in connection with relating this complex process to the mental computer. In the first place we have no experimentally verifiable reason to think that these signals, at any interesting stage of neural processing, are digital signals. They are digital, of course, with respect to the anatomical specificity of the neural pathway itself. We do not know in what way this anatomical specificity bears on the signals transmitted, except that it does provide a physical source for electrical field effects, and an origin for a variety of chemical processes. And second, we have no explicit description whatever of the relation between sensory stimuli and perceived entity. Langer has expressed this issue quite clearly.

> No matter what heights the human mind can attain, it can work only with the organs it has and the functions peculiar to them. Eyes that did not see forms could never furnish it with images; ears that

> did not hear articulated sounds could never open it to words. Sense-data, in brief, would be useless to a mind whose activity is "through and through a symbolic process, were they not par excellence receptacles of meaning. But meaning, as previous considerations have shown, accrues essentially to forms. Unless the Gestalt psychologists are right in their belief that Gestaltung is of the very nature of perception, I do not know how the hiatus between perception and conception, sense-organ and mind-organ, chaotic stimulus and logical response, is ever to be closed and welded. A mind that works primarily with meanings must have organs that supply it primarily with forms [Langer, 1942].

The structure of mental imagery has received attention from experimental psychologists in recent years, but their findings are not conclusive [Anderson, 1980], [Kosslyn, 1978]. There are propositional theorists and imagery theorists. Roughly speaking, the imagery theorist holds that the entity associated with a thought or memory is "analogue" information. There are three aspects to this assumption. One is that the mental entity belongs to a population on which a metric is definable. More exactly, these entities form a dense space. Density means that between any two arbitrarily close entities, E1 and E2, there will be a third, Em, such that Em is nearer to each of E1 and E2 than E1 is to E2. That is, the space of entities admits the specification of neighborhoods, subsets of entities which are all similar in respects determined by the way the neighborhood is specified. If a neighborhood is specified with respect to outline, colour and texture, then all objects in a sufficiently small neighborhood will have sufficiently similar outlines, colours and textures, although they may differ dramatically in other respects. Secondly, analog images are images of something else, and they are icons of what they are images of. That is, there is a structural isomorphism between the image and the thing imaged, in selected respects. However the detail of the image space is theoretically represented, small changes in the thing imaged will correlate with similar small changes in the image. Things which group into the same sensory neighborhood will group into the same neighborhood of images, relative to the appropriate choice of aspects. Finally, the analog images fetched out of memory can be processed by the same mechanisms that are recruited for processing perceptions in a similar modality.

The propositional theorist holds that these mental entities are found in discrete associative networks, in assertional or logical data bases, or in some other classically computational form. In any case, there are essential rules of well-formation associated with these structures, and the units of such structures can be described, in one or another way of reading them, as being "true" or "false" readings of reality.

Anderson has observed that it is not possible to evaluate a theory of mental representation unless that theory describes both the structures of the mental representation and the processes that will deal with those structures. [Kosslyn, 1980], make the suggestion that digital and analog structures may be processed simultaneously.

There appear to be a number of possibilities. One is that mental images are discrete and are handled by continuous or graded processes. Another is that images are iconic but are processed by digital mechanisms. A third is that mental images take on a finite and digital character as the result of complex integrative processes involving a number of distinct dense and graded substrate. This latter is not so far fetched a suggestion as it seems. Elementary particles are described by the Schroedinger wave equation as having continuously varying properties, properties which discretize to definite values when the particles interact with a suitable measuring apparatus.

In any case, the literature in experimental psychology does not justify any definite conclusion about the structure of mental entities. In particular, we do not know whether these are organized as digital structures, or as points in a continuous space.

I will make no further reference to the task-oriented approach to the study of language. We will take the framework provided by the theory of generative transformational grammar as a universal framework for the study of language, and for the study of possible ways of implicating linguistic processes in computing. The reader who is interested in such an exercise can easily verify that all studies in computational linguistics that have led to implementations exhibit in one form or another all or almost all the components of a generative transformational grammar. There are lexical items and lexical insertion rules, a categorial component or an equivalent, and a system of transformations leading to a surface structure, the form that is provided by or analyzed by the system. And in all cases there are more or less rich deep structures associated with linguistic expressions. Precise conceptual boundaries are sometimes blurred, and there are a number of techniques to make the system functional in a strong utilitarian sense. A variety of forms of knowledge bases are to be found, each with its characteristic set of functions that relate expressions to the underlying meanings that are recorded in the knowledge base.

10. Mental structures in grammar theory

It may be helpful to exploit here a certain kind of understanding of the constructions in generative transformational grammar (hereafter, 'GTG'). These remarks are subjective and motivational, and there is at present no claim that experimental verification is attainable to support them, either now or in the future. In short, all empirical, operational and utilitiarian issues are irrelevant to these remarks. On the other hand, introspection and reflection may or may not be experienced to con-

firm the validity of these observations. In an informal way, we will simply be trying to deepen our understanding of the structures and processes of natural language. I will use such phrases as "psychological reality", "significance", "relevance" and "meaning", to mention a few, without attempting definitions. The truly interesting questions about GTG are concerned with whether or not its precisely defined structures and processes confirm judgments about significance, relevance, psychological reality, meaning and other subjective notions.

The base component of a GTG is expressed in part as a finite list of context-free phrase-structure rules, whose application generates a phrase-structure tree or structural description. The individual symbols that occur in these rules are called "category" symbols. There is one distinguished category symbol which generates a primitive or elementary expression, usually a sentence. We may understand such a sentence as expressing the simplest kind of complete thought, and only the simplest kind. The category symbols that appear in these rules have one very basic function, and that is to express grammatical relations among the constituents of the complete thought. For example, if there are rules

<sentence> -> <noun phrase><verb phrase>
<verb phrase> -> <verb><noun phrase>

which yield a terminal sequence

<noun phrase> <verb> <noun phrase>

then we can say precisely that the first <noun phrase> is the subject of the <sentence>, and that the second <noun phrase> is the object of the <verb phrase>. Notice that the category <verb phrase> is abstract - it is not manifest directly in the terminal sequence. Therefore, grammatical relations may be expressed with reference to purely abstract categories. It is the phrase-structure rules which determine and express these grammatical relations, and this determination is made independent of whatever lexical items (words) are subsequently put for the <noun phrase>s and <verb>. This explanation is a psychological statement. It asserts that the constellation of objects that will figure in a simple complete thought are constrained (by the phrase-structure rules) to stand in certain grammatical relations when that thought is expressed in language. Distinct patterns of grammatical relationships will require distinct patterns of rule applications. The important point is that insofar as such rules are seen to underly linguistic expression, the thoughts so expressed are structured in part without regard to their particular content. In short, a certain structure is imposed on the simplest complete thoughts solely with reference to uninterpreted form. A category symbol has no intrinsic significance other than to indicate a class of objects each of which can stand in certain grammatical relations.

We can also see that there is no reason to suppose that a complete pattern generated by these rules needs to be generated in any particular serial order. We may even imagine that the whole pattern is generated simultaneously. We may observe that all and only the patterns generated by a context free phrase structure system can also be generated by a pushdown automaton, but for natural language we need not specify that they are so generated. So long as the immediate constituency relations are uniquely specified, we have all that is needed to determine grammatical relations. The way in which a phrase structure tree is actually generated is a matter for factual determination, and at the present time there does not seem to be any objective way to decide how a biologically normal human being generates his phrase-structure trees, indeed, even assuming that he does so.

We also see that the categorization of those entities that will be constituents of a complete thought is quite universal and comprehensive. If every well-formed uttterance has an underlying phrase-structure pattern expressing grammatical relations, then all concepts that can figure in an utterance must submit to such categorization, and this is to be independent of the content or significance of such concepts. All thought that is expressible in language derives in part from a categorial organization, an organization that expresses uninterpreted form-al relationships. These are deep and subtle observations about mental events.

Not all expressions and not all thoughts are simple. In the earliest formulations of generative transformational grammar (henceforth - GTG), this phrase-structure system had no recursive capability. In these early formulations simple thoughts were generated independently by the phrase-structure component, and were then combined by a recursive capability expressed in a system of transformational rules which took several phrase markers as arguments and yielded more complex single phrase markers as values. But by 1965 this idea was abandoned [Chomsky, 1965]. The role of the transformational component came to be concerned with the purely mechanical rearrangement of arbitrarily complex phrase markers into a surface structure suitable for articulation through the phonological component. The phrase-structure component was given just one recursive symbol, that for a simple complete sentence, so that the patterns or phrase markers generated by it might contain an arbitrary number of simple sentences. The organization of the form of a thought, simple or complex, was localized in the generation of these phrase-structure patterns, and the grammatical transformations were then understood to have little or no link with the origin of thoughts. They were, and are, exclusively concerned with the rearrangement of uninterpreted syntactic forms. The transformational component of a grammar is "autonomous and independent of meaning." The simplified picture that emerges at this stage is that thoughts are organized in categorial terms as phrase-structure patterns, in order to express grammatical relations. Transformations then apply to these patterns in order to make them articulatable according to systematic and language-specific phonological principles. And in some sense to be made precise, grammatical relations must either be

unaffected, preserved, by transformations, or else those relations that are expressed by the phrase-structure patterns must be recoverable from the surface structure.

So far we can see that both the categorial and the transformational component are concerned exclusively with the form-al manipulation of uninterpreted structures. We have given no account whatever of the introduction of specific words or word stems into linguistic structures. For this purpose a lexical component is introduced.

"A generative grammar attempts to characterize in the most neutral possible terms the knowledge of the language that provides the basis for actual use of language by a speaker-hearer." In the present discussion I understand the phrase "most neutral possible terms" to mean that grammatical description refers solely to linguistic forms and to the processes that manipulate those forms. Thus a generative grammar provides a description of the forms and form-al manipulations that a speaker-hearer uses to relate sounds with meanings. We have seen that the forms proposed by current theories of grammar include phrase markers of a variety of kinds, lexical entries, and some kind of semantic structures, and that the form-al manipulations include phrase structure rules, transformations, lexical insertion rules, rules that associate semantic structures with phrase markers or sets of phrase markers, and rules that associate sound patterns with phrase markers. All rule systems that have been proposed to associate semantic structures with phrase markers make important assumptions about the existence and form of lexical entities and about the rules which *incorporate* lexical entities into phrase markers.

11. Lexical and categorial structures

Katz and Fodor [Katz, 1963] proposed that a lexical entry was to contain a grammatical category, a set of grammatical features, a set of "semantic" features, and a distinguisher. A grammatical category is one of such as: Noun, Verb, Pronoun, etc. Grammatical features include such contrasts as count/mass and animate/inanimate, and other features indicated by such words as: young, colour, social activity, and a large number of others.

The lexical component of a grammar organizes the words of a language into classes or categories and among these categories are those that appear as terminal elements in a phrase-structure pattern. Lexical insertion is then the processwhereby words are incorporated into specified positions in a phrase-structure pattern. A phrase-structure pattern, all of whose terminal nodes are words, is a deep structure. A deep structure expresses a complete (simple or complex) thought.

The individual words may have a number of important features. Some of these concern the sound pattern that will eventually express the word. Others may concern transformations that

rearrange phrase markers containing these words. Yet others concern the possibility of inserting the word into the phrase-structure pattern in a way that satisfies specified conditions determined by the pattern.

The psychological picture here is that complete thoughts are assembled out of already clearly differentiated constituents. The mind is supposed to be in tacit command of a dictionary full of concepts and ideas, ideas which are susceptible to composition and arrangement in a way that is reflected in conventional notions about word meanings and grammatical organization. Purely extra-linguistic factors account for the specific choice of concepts to be incorporated into a complete thought, on any particular occasion of utterance.

It is clear that in certain respects a word, or lexical entry, may be described as a finite set of features, where the features themselves are chosen from a finite and definite list. In short, a word is a complex formal symbol. The admissibility or inadmissibility of a lexical item for insertion into a particular phrase-structure pattern is expressed in a completely formal way, determined by the presence or absence of certain features. The rules governing this admissibility are finite in number and are completely definite. They are computable rules. So, in fact, are all the other uses that are made of these features. At every stage of the linguistic process, beginning with lexical insertion, running through all transformations, and including the operation of phonological principles and the associated motor activity in the oral cavity, all processes are entirely mechanical, and their effects are completely predetermined. They are purely form-al manipulations.

But it is a psychological assertion that lexical items are constructed out of features. The determination whether a certain concept can or cannot stand in a specified grammatical relation in a specified context is made solely with reference to the presence or absence of a specified set of features in its associated lexical entry. From a different point of view, then, these features are simply another set of categories which condition the admissibility of a lexical entry to play certain grammatical roles in an utterance, to stand in certain conditioned grammatical relations. If a lexical entry has, say, a sense which is not expressed by its collection of features, that sense will not figure in subsequent grammatical processes. That sense can then be omitted from linguistic description without in any way affecting or altering grammatical facts. Reasonably current statements of this "interpretive" position may be found in [Fodor, 1975] and in [Katz, 1977].

It is not surprising that in recent years linguistic research has focused on the lexical component of GTGs. In 1967 Lakoff and Ross [Lakoff, 1967] suggested that the validity of the concept of deep structure could not be sustained. The generation of a phrase-structure pattern was understood to be itself intrinsically conditioned by the choice of lexical items for lexi-

cal insertion. The role of the context free phrase structure component was changed considerably as deeper complexities associated with lexical insertion were uncovered. Since the features associated with a lexical entry carried "semantic" as well as syntactic information about that entry, it was proposed that the processes that operated with these semantic features need not be distinguished in form from purely syntactic processes.

Transformations were seen to be both necessary and sufficient for all the generative processes of language, from the initial organization of thoughts in a form-al structure to the final achievement of a phonological matrix.

This "generative semantics" view proposes that grammatical processes must and do have semantic characteristics that are as important for the notion of linguistic well-formedness as are purely syntactic issues [Lakoff, 1971]. In Lakoff's presentation a lexical item has associated with it a transformation whose application to a structural description accounts for the occurrence of that item in the utterance. We notice again that insofar as all these processes are computational, or generative, they are concerned only with the manipulation of uninterpreted form.

A complete survey of current thinking about lexical and categorial structures and processes would exceed the scope and resources of this study. I will mention only superficially a few contributions in the study of these components of a grammar for a natural language.

No discussion of ideas about word meanings can be generally adequate unless it deals with such issues of word meaning as one finds in the studies of the great semanticists of an earlier period. Modern linguistic studies have tended to set these issues aside on the ground that their influence on the forms of language is not at present empirically testable. I will later propose that these influences can be accounted for by showing that the framework discussed above for the study of the human norm can be formalized at least as much as can the framework for generative grammar. Specifically I will propose definite shapes for the mental forms associated with lexical entities, transformational rules, structural descriptions and the agencies that administer such processes as the transformational cycle. But at this point I will simply reproduce a few selected observations from Stephen Ullmann's classic study of semantics [Ullman, 1951]. These observations provide subtle and powerful insights into language meaning, insights which are easily lost in the face of an excessive and exclusive preoccupation with formal manipulation.

"Language has various means at its disposal to convey vagueness, uncertainty, approximateness, the lack of sharp contours. Expressions like 'about a hundred', 'greenish', 'not unlike', 'comparatively', are referentially designed to introduce this element into discourse. These explicit devices must be kept apart from the implicit vagueness which is the most striking differentia of the sense.

"The vagueness of the sense is an inherent but highly variable feature. It is a consequence of the process of abstraction by which our 'concepts' are evolved. Without going into psychological details, it is clear that even the reference called up by proper names is a mere 'schema': what we think of when we hear the name 'Napolean' is a telescoping of the artillery officer at Toulon, the victor of Austerlitz, the exile of Saint Helena, and so on... With generic names the simplification and reduction to a bare outline, or even to a mere 'act of reference' devoid of any perceptual elements, becomes more marked, and the gap between the virtual sense in the language system and the actualized sense of speech-contexts widens considerably.

"There are also more specific reasons for semantic vagueness. In some cases, lack of sharp demarcation lines is a property of the referent itself. The difficulty of delimiting the various parts of the human body is thus responsible for the shift in Latin 'coxa' 'hip' > French 'cuisse' 'thigh'.

"When assessing the emotive components of the semantic relationship, the term 'emotive' must be taken as a kind of pars pro toto for all non-cognitive factors entering into verbal configurations. It should thus include affective as well as volitional aspects, and, from another angle, both expressions of one's own feelings and the arousing of feelings in other people.

"The affective side of language is just as fundamental as its cognitive function. In theory, every utterance is both communicative and emotive: there is always something to be said, and a subjective interest in saying it. ... The two elements are in principle always compresent in speech; it is only their dosage that varies.

"It is quite natural for all elements of the language system to play their part in discharging so fundamental a function. Emotions can be conveyed in many ways: by intonation and rhythm, by the choice of suffixes (endearing, pejorative, etc.), by word-order and syntactic arrangement, etc. It was realized by semanticists at an early stage that no complete description of the sense of a word can leave out of account its affective 'overtones'.

"... in most cases the situation alone can tell whether a term is used referentially or affectively... Even the most prosaic objects can suddenly acquire unexpected sentimental overtones:

> Thou wall, O wall, O sweet and lovely wall,
> Show me thy chink, to blink through with mine eyne!
> Thanks, courteous wall: Jove shield thee well for this!
> But what see I? No Thisby do I see.
> O wicked wall, through whom I see no bliss!
> Cursed be thy stones for thus deceiving me!

"... some emotional elements are neither individual nor purely contextual in character: they are a permanent accompaniment of the word and sometimes its very raison d'etre. ...girl - lass - maiden, mother - mummy - mater, little - small - tiny - teeny - wee. ...

"Many terms of praise and reprobation become saturated with the moods and feelings attaching to them in innumerable contexts: 'freedom', 'liberty', 'democracy', 'right', 'persecution', 'tyranny' - all preferably with a capital initial - as well as the various -isms and other fashionable slogans are examples in point.

"The stylistic nuance of 'barbarisms, provincialisms, vulgarisms, archaisms' and technical terms lies outside the scope of semantics; much of it belongs obviously to 'la parole' and not 'la langue'. Emotive undertones of this kinds may, however, have more permanent effects; they may provide the momentum which carries a word outside its linguistic, dialectal or social boundaries, and which occasionally sets off major processes of linguistic borrowing altering the whole structure and fabric of the vocabulary.

"Recent investigations ... have discovered close parallelism between conventionality and expressiveness on the one hand, referential and emotive meaning ('notional' and 'interjectional' meaning) on the other. The more affective the import of a word - either permanently or in a given context - the more alive we become to its expressive resources, and vice versa. A sliding scale stretching from scientific terms at one end to pure interjections at the other gives an idea of the varying utilisation of evocatory facilities.

"Another feature which emotive power has in common with the kindred phenomenon of motivation is its subjective validity. Verbal likes and dislikes are notoriously unaccountable, and one person may find comical what another has found expressive or moving. ... Where Wordsworth writes: 'And sitting on the grass partook the fragrant beverage drawn from China's herb,' Tennyson, in his customary bias against non-Saxon words, comments: 'Why could he not have said "And sitting on the grass had tea."?'

"Poetry is the medium par excellence of emotive meaning, and it is by no means surprising that a purely cognitive approach to its 'message' may frequently lead to misunderstandings, or to utter bewilderment."

I will not pause to comment on Ullmann's observations here, except to note that the emotive content of words is as "fundamental" and significant as is their cognitive content, and to point again to the first quoted observation about the vagueness and approximateness that appears to be a fundamental part of

language. An undue fascination with the purely formal manipulations of language can easily hide these and other significant properties of language from our view.

We have already discussed the semantic theory of Fodor and Katz. Some of the later developments in lexical and semantic theory were criticisms of parts of this theory, and others have presented more comprehensive alternatives.

Weinreich's principle criticism of the Fodor-Katz theory was essentially that the relationship of semantic features was not explicitly represented with reference to the grammatical relationships that were determined in the deep structure [Weinreich, 1980]. In part to correct this defect Weinreich proposed that bundles of features had to have some systematic organization imposed on them. For example, he distinguished between a "cluster" and a "configuration" of semantic features. A cluster is simply an unordered set. But the elements of a configuration are linearly ordered, where this ordering expresses a certain conditional relation between semantic features. For example, a daughter is a female and an offspring, and these two features make up a cluster, because, in the meaning of the word 'daughter' the contributions of 'female' and 'offspring' are independent of each other. On the other hand a chair is involved with 'furniture' and with 'sitting', but only in a conditional way. To paraphrase in pseudo-language, 'female' can 'offspring' and 'offspring' can 'female', and in both cases we get 'daughter'. "Furniture' can 'sitting', but whatever this means, we cannot have in the same sense that 'sitting' can 'furniture'.

Weinreich then specifies two operations, one which produces composite clusters and another which produces a family of different kinds of composite configurations.

He also proposes an optional second kind of ordering of the elements in a cluster, one in which earlier features in the cluster would have more prominence in the utterance, more emphasis, than later features. Without such a convention, for example, the sentence

A small elephant is big.

would be interpreted to be contradictory.

A further interesting idea is that of a "transfer" feature. To say of a craft that one sailed it is to have the transfer of a 'water vehicle' feature from 'sail' to 'craft'. To say of the craft that one flew it is to transfer the feature 'airplane', or some similar feature.

Fillmore has made more radical proposals for the lexical component and the categorial system [Fillmore, 1968, 1972]. He describes a lexicon in this way.

"A lexicon viewed as part of the apparatus of a generative grammar must make accessible to its users, for each lexical item,

(i) the nature of the deep-structure syntactic environments into which the item may be inserted;

(ii) the properties of the item to which the rules of grammar are sensitive;

(iii) for an item that can be used as a 'predicate', the number of 'arguments' that it conceptually requires;

(iv) the role(s) which each argument plays in the situation which the item, as predicate, can be used to indicate;

(v) the presuppositions or 'happiness conditions' for the use of the item, the conditions which must be satisfied in order for the item to be used 'aptly';

(vi) the nature of the conceptual or morphological relatedness of the item to other items in the lexicon;

(vii) its meaning; and

(viii) the phonological or orthographic shapes which the item assumes under given grammatical conditions.

The roles which arguments play in a situation indicated by a predicate are formalized in Fillmore's treatment of grammatical "cases". He specifies a partial list of cases in this way:

Agent (A), the instigator of the event

Counter-agente (C), the force or resistance against which the action is carried out

Object (O), The entity that moves or changes or whose position or existence is in consideration

Result (R), the entity that comes into existence as a result of the action

Instrument (I), the stimulus or immediate physical cause of an event

Source (S), the place from which something moves

Goal (G) the place to which something moves

Experiencer (E), the entity which receives or accepts or experiences or undergoes the effect of an action.

For example, the sentence

Eddie gave the yogurt to his doggie.

will originate from a deep structure pattern which, prior to lexical insertion will consist of a verb and a set of noun phrases each of which is identified or labelled for a specific case relation to the whole sentence. For this example 'give' will be the verb, 'Eddie' will be the Agent, 'the yogurt' will be the Object and 'his doggie' will be the Experiencer. One could of course express all this as a labelled tree structure, but such structures will be purely notational devices for Fillmore since each of the constituents in deep structure is already identified for function, and this identification is itself adequate for at least part of the structural description. Specific rules for introducing prepositions will operate upon the identification of cases. The Agent preposition is 'by'; the Instrument preposition is 'by' if there is no Agent, and otherwise is 'with'; the Experiencer preposition is typically 'to'; and verbs of motion will have other specific prepositional rules associated with their various cases, and will also condition the prepositions used in connection with reporting various forms of motion.

Transformations applied to this deep case structure will then be capable of accounting for any of

Eddie gave his doggie the yogurt
His doggie was given the yogurt by Eddie
Eddie's giving his doggie the yogurt...
Did Eddie give his doggie the yogurt?
What did Eddie give his doggie?

and others. Fillmore goes on to consider in some detail the mechanisms by which grammatical subjects and objects are determined.

Gruber takes a different approach to the form of lexical entries and to the forms that the categorial component must generate [Gruber, 1976]. He cites example like the following.

Wally wants some yogurt.
Wally yearns for some yogurt.

He observes then that these sentences must share certain essential structural features in deep structure, say

(<noun phrase> (<verb phrase> <noun Phrase>))

The preposition is sometimes part of the verb and sometimes not. We do not have

Wally wants for some yogurt.
Wally yearns some yogurt.

although we can easily interpret at least the first of these "deviant" sentences. Other verbs make some prepositions optional.

Eddie strayed from the path of virtue.
Eddie strayed away from the path of virtue.

We can have

Eddie strayed.
Eddie strayed away.

but we cannot have

Eddie strayed the path of virtue.
Eddie strayed away from.

The process whereby the preposition is not expressed in the surface structure but is included with the "meaning" of the verb is called "incorporation", and Gruber finds wide application for the notion. He finds evidence that complete noun phrases are subject to incorporation.

The cat is eating tuna.
The cat is eating string.

But when we hear

The cat is eating.

unless other factors tell us to expect facetiousness, we assume that it is some kind of food the cat is eating and not some kind of string.

Adjectives can also be incorporated.

Your casserole smells delicious.
Your casserole smells disgusting.

But for the sentence

Your casserole smells.

we normally expect that a certain unpleasantness is alluded to.

To accommodate the formal manipulations associated with this range of phenomena Gruber proposes that a lexical entry is expressed properly as a phrase-structure exhibiting those elements that may or may not be incorporated in the surface structure. For the verb 'escape' we have something like the following in the lexicon:

(MOTIONAL POSITIONAL :[[OUT-OF] <noun phrase>])

where MOTIONALand POSITIONAL are features that identify the character of the verb, and OUT-OF is a semantic item that represents the part of the meaning of the verb that may be expressed prepositionally. The square brackets indicate optionality, so that we can account for all of

Wally escaped.
Wally escaped the doghouse.
Wally escaped from the doghouse.

The base component is then required to generate structural descriptions that can accommodate the introduction of prepositions where necessary, or their omission when the lexical structure represents that the preposition may be incorporated or must be incorporated.

For example, verbs involving a goal may have, in their lexical entry, a semantic item TO, which may be expressed by different prepositions for different verbs. Categorial structure would be necessary to mediate these. Here are some examples of goal sentences which are expressed with different prepositions.

Eddie ran below the boardwalk.
The doggie scooted in front of the building.
Wilbur jumped on the truck.
Birdie flew into the birdbath.
She ate off the tray.
He rolled the hoop ahead of him.
He rolled the hoop behind him.
He rolled the hoop in front of Wallis.

In earlier treatments of the lexical component, lexical items were attached to a single node in the phrase structure tree generated by the base component. For Gruber, this attachment may involve several nodes of the categorial tree, and may even involve some of the phrase structure. These moves are needed in order to handle the prepositions, and the general phenomenon of incorporation. It requires complications to the categorial system, and a fundamentally more complex and more powerful lexical attachment process. Briefly, the process whereby the components of a simple primitive thought are assembled together for utterance are conditioned by the way in which the thought will be expressed in surface structure.

I will not review any further work dealing with categorial or lexical structures. Instead, we turn now to a suggestion of a new approach to the description of linguistic and other mental structures, processes and rules.

12. The form of meaning

A. Preliminaries to formulation

In this section we propose an alternative to the structures and processes that are currently involved in the study of language and the semantics underlying language. These ideas have been heavily influenced by G. Spencer Brown's study of form, [Brown, 1972], but I will not discuss in any detail here the exact relation between Brown's ideas and those given here. I will only mention that he takes the primitive concept to be that

of drawing a distinction, and he proposed to explain all the methods of construction in coherent articulation with reference to this notion alone. His work when approached seriously makes it quite clear that this very simple but quite obscure idea is at work at every turn in reasoning and speaking. His conclusion, for example, is that the processes of drawing distinctions are the same whether we fix attention on the marking of a distinction, the calling of it into awareness, the movement across the boundary of the distinction, or the observation of it.

The alternative meaning structures proposed here are based on a number of assumptions.

(1) In the absence of any conclusive evidence to the contrary, it is inappropriate to ignore the possibility that the mental and physical structures and processes associated with the human norms's behavior are in important respects distinct.

(2) Mental structures and processes are orderly, and this orderliness may be expressed in a definite way as a system of structures and processes which are common to all "normal" human beings.

(3) These common structures are concerned with the manipulation of forms - abstract and uninterpreted computational structures.

(4) Individual humans deviate from this common norm in the particular uses they make of these forms, and in constrained but characteristic variations of these forms. These particular uses are themselves determined by the individual's experience (past and present), by a personality specific pattern of preferential judgments and by a system of conceptualizations.

(5) These forms are the carriers of awareness of perception, of feeling and of emotion, but are never themselves direct participants in acts of awareness, perception, feeling or emotion. They are thus not observable by introspection, and insofar as they are independent of physical structures and processes, neither are they observable by physical instruments. In short, computation is the unobservable carrier of conscious experience.

In spite of (5) I think it is possible that we may infer something about the forms of mental structures and processes indirectly by abstracting from the behavior of human beings in a variety of different kinds of situations. I think that the ideas presented here may make it possible to pose empirical questions about the nature of these forms, questions whose answers may be obtained by experimental procedures. It is also possible that these ideas may lead to useful technological and methodological applications.

Specifically the proposal is that all mental entities which are carriers of experience are recurrences. These recurrences will exhibit regularities in their forms. That is, they will combine together in orderly ways to make composite recurrences, and composite recurrences will be subject to decomposition into constituents. The intuitive notion of recurrence is that something "occurs" again and again without so much change that we cannot interpret it as a re-occurrence, a recurrence. For example, it is a mental process to hold onto a memory of an occurrence of a sensation after the sensation itself has come to an end. The basis for perception is not occurrences of sensations but recurrences of sensation-traces. And it is those recurrences that are subject to abstraction and elaborate conceptual organization. A perception is itself carried by a concept-conditioned form which recurs through time. A feeling or an emotion is the application of a preferential judgment applied to a form, and is itself also carried by a form that endures through time. If a thought has propositional content, that content is carried by a concept-conditioned form that recurs through time.

An irreducible recurring form will be one whose composition structure if it has any, does not carry any relevant aspect of experience. Irreducibility is thus itself a relative notion. And indeed, we may speak of reducibles in place of composites, understanding the same relativity. What is irreducible in one situation may be composite in another.

I gave my daughters dolls and my sons robots.
I am meeting my daughter for lunch.

If, in feature terms, we think that daughter has the features [offspring] and [female], then those distinct features are relevant for the first sentence but not for the second. Perhaps the only feature that might, in context, be relevant, would be that of [relationship].

The recurrence of a reducible form is in general subject to indeterminacy. A variety of factors influence the degree of indeterminacy, but in general it cannot be eliminated for arbitrary lengths of time, for arbitrarily extended durations of the recurrence. We do not need many examples of inherent vagueness in language.

People never say things like that.
Boys naturally like football.
Her singing was impeccable.

For computer realization, the presence of indeterminacy indicates that the computer system will be an idealization. We note that according to these ideas it is the form itself that is indeterminate and that as a consequence the experience carried by it will be indefinite or vague.

Recurring forms cannot themselves be directly experienced. We might anticipate then that composition operations involving them will not be altogether familiar. Three such operations will be described here, although we may eventually postulate a set of operations that are themselves specified with reference to a system of meta-operations. The three we describe here are fusion, coordination and subordination. Fusion is an operation which combines two (recurring) forms into one. One speaks of a "fusion event". Fusion "blends" its constituents together. Those constituents are then not directly available in the composite. Fusion, called '*', has the property that although z is the result of the fusion event taking place with x and y, we may not be able to recover x and y as the unique constituents of that fusion. Our instincts for rigour may be offended by an operation like fusion, but it is too bad. Life is like that.

For example, [featherless * biped] may yield [human], but we may later analyze [human] into [mammal * with humor]. Here we have a first halting step toward representing Michael Polanyi's "tacit" knowledge. And it also provides us with a way of representing the difference between background and foreground knowledge. Fusion tends to produce irreducibles.

The difference between fusion and coordination is that the constituents of a coordination are always recoverable in a definite and unique way. There is no other difference. Like fusion, coordination is associative and commutative and idempotent. One reason to have this contrasting pair of combinations is to account for the fact that information can be relevant for a time and then may become irrelevant. Thus a coordination may turn into a fusion, under specified conditions. Depending upon the perspective we may describe the process of turning a coordination into a fusion as a simplification or as a generalization. And the inverse process of turning a fusion into a coordination as an elaboration or a refinement. For example, the process whereby the phrase "song and dance" comes to express a unitary idea is an instance of a coordination becoming a fusion. All events of composition in which constituents retain some structural integrity will involve coordination. These are widespread. "Juicy red apple", "red hot momma", "Bring the knife, the butter and the toast!".

Fusion is as widespread as coordination, but it is less easy to recognize because it marks the boundary at which distinctions become inarticulatable. In many cases this is also the boundary where what is explicitly grammatical becomes what is colloquial. Such phrases as "here and there", "now and then" and "this and that" are typical. But fusions may participate along with other irreducibles in clearly articulated grammatical constructions.

His song and dance no longer fools us.
Run this over to the Dean's office.
She'll bend your ear all afternoon.
Give us a break!

But since fusion tends to produce irreducibles, we do not in general expect to see composites of any detail with interesting fusion characteristics.

Fusion may also turn into coordination, but with results that are somewhat indeterminate. The transformation from fusion to coordination may produce information that was not present in the "original" coordinate, or it may leave out information that was originally present. It is possible that this transformation is related to the processes whereby information is integrated into memory or retrieved from memory, processes that are surely conditioned by a number of distinct factors and influences. Fusion may also be related to the processes attendant upon shifting attention, sharpening or diffusing attention, or expanding and contracting the field of awareness. Coordination is called '.'. For example, [juicy . red . apple].

Subordination is quite routine. It establishes a definite precedence between its constituents. This abstract precedence is the origin of both sequential order and hierarchical order, depending on the structure already present in the constituents. To keep notation uniform we will indicate that x is subordinate to y by writing either of

$$x(y) \qquad y(x)$$

We will also be interested in the "transitive closure" of the operation of subordination. There is an ambiguity of interpretation which we will not sort out here. Thus 'y(x)' indicates either the result of the composition event of subordination on x and y in a specified order, or else it indicates the fact that in the appropriate setting x is subordinate to y. Then the transitive closure of subordination can be specified more precisely by the conditions

$$x(y) \rightarrow x[y]$$
$$x[y] \text{ and } y[z] \rightarrow x[z]$$

Here the pattern ...[...] indicates the transitive closure of the relation indicated by the pattern ...(...).

For some x it may be the case that x[x]. In such a case x may be either reducible or irreducible. If x is irreducible and if x[x] then we say that x is a recurrence and also that it is a recurrence which recurs in discrete and distinct instances. If x is reducible and x[x] then we may take that as the condition that a composite be a recurrence, and that it recurs in distinct consecutive occurrences.

In connection with these ideas, which we must set aside for now, there are clearly questions of well-definedness, consistency and so on. How do these operations act when they are applied in succession? Are there any new issues that require systematic treatment when we speak of events of combination in place of the

more usual result of combination? In what logically specified framework will we write the axioms that determine the effects of these operations?

We may pause for a few more simple examples and an indication of how these ideas might be applied. It is clear from a careful reading of the literature in linguistics that the notion of composition is far from clear. Interesting ambiguities are present and absent in contrasting situations that bear superficial similarities. Consider, for example, these phrases.

The prattling of the gossip
The eating of the meal
The shooting of the hunter

The issue in language study is to understand in what way the form of an expression can admit ambiguity in one case and disallow it in another.

Or consider

Big European butterflies
The little elephant

In what way does the form of the expression, for example, relativize the concept 'little' to the concept 'elephant'? In what way does this form carry the relativization of 'big and 'European' to 'butterfly' at the same time that it deals with the ambiguity between [big [European butterfly]] and [[big European] butterfly]? Are these clear instances of subordination in some kind of abstract representation? Does a well-formed expression in a natural language, for example, have a number of distinct structural representations all held simultaneously while the deep structure is present in some as yet unspecified sense in consciousness?

How exactly does scope of awareness or of attention figure in the structure of an expression? Is it possible that the structure of an expression is undergoing continuous change in a systematic way, throughout its presence in a field of awareness or a field of relevance? Grammatical transformations have been adduced to account for the relations among different forms of the same deep structure, or for the relations between simpler modifications of the same deep structure. Such examples as the following were cited.

Eddie gave his doggie some yogurt.
Eddie's giving his doggie some yogurt.
Eddie's doggie was given some yogurt.

How do we account for the relationships that obviously hold between consecutive pairs in this example, and thus between any two in the set?

It was a noble thing that Eddie gave his doggie some yogurt.
It was a noble thing that Eddie did to his doggie.
It was a noble thing that Eddie did with some yogurt.
It was a noble thing that Eddie did.
Eddie did something noble.
Eddie did something to his doggie.
Eddie did something.

Surely the relations among these distinct utterances are not arbitrary. In one direction there is an increasing simplification or generalization with a corresponding loss of specificity, and in the other direction there is an increasing detail or refinement or increase of definiteness. We have seen this same process in connection with irreducibles and with colloquial usage.

How is this issue related to that most important capacity in language, the ability to form metaphors? If A is a metaphor for B how do we account for it structurally? Do we say that in certain interesting respects we ignore some of the detail in A and some of the detail in B, and then observe that if thes ignored details are judiciously selected the resulting simpler forms will be the same? Or is it the case that what is ignored must have the same form, and what is left behind in each case is distinct?

These are fundamental and important questions about language, and we expect to attempt tentative answers for them in the future.

B. Background

Recurrent phenomena occur in a number of different situations involving mind. And it is suggested that recurrent structures will account naturally for many aspects of mental phenomena that may require explanation.

For example, we may examine structures in the nervous system that seem to be directly implicated with behavior organized in terms of mentally conceived abstractions. The reflex arc is a cyclic or recurring event. Autonomic processes, insofar as they are understood at all, are clearly mediated by closed neural structures that realize or implement recurrences. (A comprehensive source for questions of fact about neurophysiology is [Mountcastle, 1974]. While it is six years old, I do not think that any relevant facts have been superseded.) A particularly interesting example of recurrent cooperating parallel processes is given in a recent report on the neural generation of swimming movement in the leech [Stent, 1978]. As a result of a comprehensive survey of the neurons in adjacent segments of the leech, the assembly of neurons and their pattern of interconnection has been recognized that accounts for the "sine wave" pattern of the leech's swimming movement. The reality here is a system of parallel processes that are quite precisely integrated. In detail this integration is achieved by neural interconnections of both

excitatory and inhibitory synapses. But at a suitable and significant level of abstraction (a level that in fact ignores the actual movement itself) the plan, the form of the phenomenon, is expressed as a system of cooperating parallel processes. The transformational cycle in generative grammar is just that - a process that must be prepared to operate an arbitrary number of times at successive levels of nesting in a complex expression. Retrieval from memory is often described as a process involving recruitment, and the engagement of neural substructures can, in a quite well-defined way, be described as an entrainment phenomenon. One suspects that given a suitable definition of the form of a memory, certain aspects of recruitment could be described as an entrainment process.

The experience of music, not only at the physiological level but as well at the level of abstract concepts is decisively a matter of recurrent processes and entrainment. At a fine level of detail, periodic phenomena account for pitch and for differences in pitch. The experience of a single period does not produce a sensation of pitch. At the other end of the scale periodic phenomena account for what we experience as rhythm. Metric structure is the skeleton with which harmonic and contrapuntal elaborations are integrated. Harmony itself is a recurrent matter both in its internal structure and in regular recurrence of harmonic patterns so much favored in so many distinct kinds of music. And one could hardly question the periodic nature of contrapuntal development.

Thoughtful reflection will reveal that the notion of a system of precisely integrated parallel processes is almost comprehensive. Further expansion of these purely suggestive remarks would be of no value. But future research will consist in part of simulating a range of simple and basic phenomena related to behavior to show just what is entailed by representing them as processes subject to entrainment rather than as static entities subject to fixed structural relationships.

C. Occurrences

At an earlier time in this study we spent some considerable effort attempting to understand the concept of an occurrence. That study was in large part motivated by a perhaps naive grasp of the use of the notion of a well-formed formula in a formal language. In that little study an effort was made to achieve some kind of useful processing of language text without the need to attribute meanings to its parts. The approach was itself misguided by an excess of respect for static structures. I think that the reality is that expressions in language are what are called 'epiphenomena'. (I do not mean to disparage the expression by this remark. How can we disparage an expression in a language. It is everything we find it to be!)

Given two theories, one which postulates a lot of entities and another which postulates just a few, we seem to prefer the more sparse ontology, other things being equal. We propose that

other things can indeed be kept equal while postulating nothing but recurring, cooperating processes. Whether this is or is not the case will be determined by future investigations. The remarks we reported earlier concerning occurrences are included as Appendix A at the end of this report. The results there were inconclusive because we were not able to relate the notion of occurrence to the entities that occur. In the treatment given here we will understand that it is processes that occur, and this should eventually clarify our grasp of the way in which processes in thinking and speaking interact in a coordinated way, according to rule.

D. General Implementation Notions

We are interested in designing a computational system twhose central ideas are those involved with the integration of parallel processes. Individual processes must be able to direct any conventional computation, but there must be a rich set of facilities for managing the integration and cooperation of an arbitrary number of simultaneous processes. The ability to call on LISP (or indeed, in other possible implementations on any high level language) is as we shall see quite easy to provide. Here we want to describe the integrating or interacting facilities that are desirable in a system oriented toward parallelism as a high level phenomenon.

(1) Individual processes must be capable of being in a number of distinct "states of arousal". If a process is undergoing transition between successive configurations, then it is an ACTIVE process. A process may be SUSPENDED in which case it undergoes no transition unless its state of arousal is changed from outside. A process may be in a WAITING state, and will then have associated with that state a waiting condition. A WAITING process is regularly interrogated to determine if its waiting condition is satisfied, and when it is, the process is brought to an ACTIVE state of arousal. A process may reach a configuration that is a natural ending configuration. Then it is in a TERMINAL state. Terminal states are of two kinds, following a policy that governs termination of finite state processes. There are really two kinds of terminal states, TERMINALACCEPT and TERMINALFAIL. These two terminal configurations are distinguished by the set of transition specifications that controls the process itself.

(2) A process should be able to initiate or terminate any other process in the system. It should be able to change the state of arousal of any process in the system including its own. It should be able to alter its own or any other wait condition freely, unless specific indicators forbid it from doing so.

(3) A process should be able to replicate itself exactly, or with arbitrary modifications, so that more than one identical copy of a process may be active at any given

time. s 1. (4) With each process there should be a protected set of identifiers determining a local environment for the process's activities. There should also be a background environment that is freely accessible by all processes for any reason. The local identifiers unique to a particular process ought, however, to be available to other processes for inspection, if not for modification. There should be no artificial restrictions on the values associated with registers when a process is created or initiated, unless such restrictions are explicitly stated.

(5) In execution it must be possible to specify interrupting situations as freely and arbitrarily as the user might want. Some processes may honor interrupt mechanisms and other processes may, at the user's specification choose to ignore them. Or the honoring of interrupt states might be made contingent on the configuration the process is in when the interrupt mechanism interrogates it.

(6) At interrupt time the user should be able to examine all details of all and any processes in the system. He should be able to make any changes he wishes at this time - initiating new processes, changing the configuration or state of any process, deleting processes, etc.

(7) Individual processes must be uniquely identified by name, and must be suitable objects of other processes. We have then the unusual situation that a dynamic ongoing computation is itself the value of an identifier and is subject to manipulation in suitable ways. At the present time it is not clear what we will take as selectors and constructors for a computation, but it is clear that the concept of a predicate that takes computations as arguments is a quite reasonable kind of thing.

(8) Every computational system has the capacity to ask questions and choose alternative sequels depending on the answer to the question. Question asking must, in this system, be the most powerful facility available. This means that a full mechanical deduction system must be available for these purposes. It must be possible to carry out these deductions on information in the background environment, in the local registers, or in mass storage. It must also be possible to reason about the processes themselves, about their interior structure, about their past history and in particular about their integration with other processes.

(9) All facilities made available to the system must be incorporatable into the processes themselves so that one has full and complete choice whether to run in a heavily interactive mode or in a fully automatic mode.

In the next section of this report we will describe in detail the prototype implementation in LISP of a primitive system for mana-

ging the simulation of parallel computations.

13. Integrated Parallel Processing

A. Introduction

In this section we describe a facility which, when added to a high-level programming language, provides a general facility for specifying the details of the temporal organization of a system of parallel computations. We take the point of view that in some cases an interesting computation can be specified in terms of a number of parallel computations that interact with each other. In this particular study we have embedded this facility in the LISP system as modified at the University of California at Irvine, and subsequently at Rutgers University. Computational steps that are not germane to the control structure processes introduced here thus may be of any size or complexity. In particular, these steps may be any that are obtainable as the result of submitting an S-expression to the LISP evaluator. As we shall see, however, there is no particular reason to imagine that these facilities are best implemented in a LISP context. Aside from some questions of programming convenience, these facilities could be embedded in any programming language.

The basic elements in the programming process described here are quadruples and sets of quadruples. A quadruple is a list of four things, much like the quadruple for a Turing machine. The first element of this list is a "current state" and its last element is a "next state". But we depart sharply from Turing's original formulationand permit any S-expression to occur as the second element of a quadruple. Following the LISP convention, this second element, or "test", succeeds if its value is anything other than NIL, and it fails if its value is NIL. In a similar way the "act" or third part of the quadruple can also be any S-expression. Ordinarily we envision that a call on the act part of a quadruple will initiate a process having substantial "side effects". A set of quadruples thus specifies a system of possible transitions from state to state, conditioned by tests expressed as LISP S-expressions, and having environmental effects mediated by the evaluation of other LISP S-expressions.

It is explicitly allowed that the part of the computation specified by these sets of quadruples may be non-deterministic. A process which begins as one computation may thus expand at non-deterministic points into a family of computations. The facilities we describe here permit communication among all these computations, and permit a computation to intervene, even decisively, in the course of another computation.

I call a computation a "world line". We may summarize the new facilities as follows. A world line is initiated by specifying its environment, its initial state, and certain control flags. One world line may initiate,terminate change or may interrogate another. One world line may suspend another, and a suspended world line may only become active again by the action of some already active world line. One world line may set an-

other world line waiting. A world line which is waiting may become active by having an associated wait condition evaluate to something other than NIL. Thus, a world line may be made inactive by the action of another world line, but its restoration to active status may be effected either as a consequence of the evaluation of its own wait condition or by the activity of another world line. By referring to the history of a world line, its ancestor world lines, one world line may act upon all the surviving descendents of a specified world line.

A world line dies when it cannot achieve transition into another state due to the fact that there is no available quadruple with that world line's current state as its first member. Such world lines are saved in a list of dead worlds. A world line which cannot transition because its test fails is annihilated. Thus a world line may be active, suspended, waiting or dead. Only active world lines try to transition. But at the end of each sweep of all active world lines, a sweep of all waiting world lines is conducted during which each waiting world line's wait condition is evaluated. The result of this evaluation determines whether the waiting world line is made active again.

There is also an interrupt facility which suspends the transitioning of world lines so that the user may intervene, inspect the state of the processes, and make changes.

Another peculiar facility is provided in connection with a world line's environment. Although the computation that proceeds during the elaboration of a world line may deal with a general LISP environment, state transitions for a world line are ordinarily effected with reference to a fixed set of registers, whose status is best described as follows. The tests and acts in a quadruple may make reference to any of a number of registers. A complex test may make changes to these registers. If the test succeeds, then the registers are treated as though their contents had been called "by name", but if the test fails, then those contents are treated as though they had been called "by value". Thus, in the presence of a failing test, changes to the registers are ignored, while successful tests make such changes permanent.

In connection with the interrupt facility one may suspend or activate <world line>s or set <world line>s waiting, where this activity is contingent upon a <world line>'s having a specified <world line> among its <ancestors>. It may also be contingent upon the success or failure resulting from the evaluation of a specified condition. The evaluation of this condition may refer to the <world line>'s registers and the user may specify that registers are to be called by name or by value, when this condition is evaluated.

In subsequent parts of this section we describe the LISP functions which provide an operational implementation. Then in the next to the last section we describe some extremely simple test cases which verify at an intuitive level that the implementation operates as advertised, and in the last section we

discuss the application of this programming methodology to some specific computational problems, and sketch the next steps in the project.

There is no literature review here since these developments have come too recently to permit the very specific kind of search that is required. And the programming facilities provided here are hardly offerred as a well-defined new programming language. But whatever appropriate language constructs may evolve, they will make reference to the facilities described here. In a fashionable jargon, the semantics is provided but not the syntax.

Finally, it should be noted that the facilities are programmed in a raw form. The construction and reconstruction of world lines is done here using commonplace LISP constructors. An efficient implementation would certainly take the view that its function is to arrange and rearrange pointer structures, obviating the need to assemble new lists at every stage of the process. And extensive error-checking facilities should also be provided.

We have not begun to address the question of verification of processes specified in this system. In particular we have not treated explicitly the details surrounding points of contact between two processes. These matters are usually discussed with reference to such notions as semaphores, critical regions and others. [Brinch Hansen, 1973] provides an adequate introduction to these ideas, but much has occurred in this rapidly expanding field. Eventually it will be important to place the present work in proper relation to the ongoing studies in operating system design and function.

I am presently calling this set of facilities KOTSU. A KOTSU is a stick used in Zen monasteries to make a point, to sit on, or to rap students who do not remain upright. A KOTSU is about fifteen inches long and is shaped like the human spine.

A. Computational Environments

Processing takes place with foreground and background information. The background is completely unprotected. Any activity whatever that references or changes the background does so in an unrestricted way. Therefore, processing stages that fail may leave a record or trace of that failure in the background. The foreground is, in a well-defined sense, protected. Certain processes have a tentative status, and until that status is resolved, associated changes to the foreground are provisional. But the resolution of the status of a process may make changes to the foreground permanent.

The foreground is implemented as a set of registers or attention fixers. There is theoretically an unlimited number of these registers. They are systematically named: *1, *2, *3, ... A special register, *0, is available, in an unprotected way, to all processes. The background is the LISP environment.

C. Quad Sets

An algorithm is expressed as a <quad set>, a set of quadruples.

```
<quad set> ::= (<quad set name> <start state>
                  <register list> <quad list>)
```

Here are the selector functions for <quad set>s.

```
(DEFPROP  QUADSETNAME (LAMBDA (X) (CAR X)) EXPR)
(DEFPROP  STARTSTATE (LAMBDA (X) (CADR X)) EXPR)
(DEFPROP  REGISTERLIST (LAMBDA (X) (CADDR X)) EXPR)
(DEFPROP  QUADLIST (LAMBDA (X) (CADDDR X)) EXPR)
```

CHANGESTARTSTATE is a service function which changes the <start state> in a <quad set>. Following it is a function which changes the <quad set name> in a <quad set>. This latter function also changes the names of all the <quad>s in the <quad set>. Each of these functions returns the changed <quad set>. A <quad> is defined just below.

```
(DEFPROP CHANGEQUADSETNAME
 (LAMBDA (QUADSET NEWNAME)
  (LIST NEWNAME
        ( STARTSTATE QUADSET)
        ( REGISTERLIST QUADSET)
        (CHANGEQUADSNAME QUADSET NEWNAME)))
 EXPR)
(DEFPROP CHANGESTARTSTATE
 (LAMBDA (QUADSET NEWSTARTSTATE)
  (LIST ( QUADSETNAME QUADSET)
        NEWSTARTSTATE
        ( REGISTERLIST QUADSET)
        ( QUADLIST QUADSET)))
 EXPR)
(DEFPROP CHANGEQUADSNAME
 (LAMBDA (QUADLIST NEWNAME)
  (MAPCAR (FUNCTION
            (LAMBDA (X)
             (LIST ( CURRENTSTATE X)
                   ( TEST X)
                   ( ACT X)
                   ( NEXTSTATE X)
                   (COND [(FULLQUADNAME? ( QUADNAME X))
                          (CONS ( SETPARTOFFULLNAME X)
                                NEWNAME)]
                         [NEWNAME])))))))
 EXPR)
(DEFPROP CHANGEQUADSETREGLIST
 (LAMBDA (QUADSET REGISTERS)
  (LIST ( QUADSETNAME QUADSET)
        ( STARTSTATE QUADSET)
        REGISTERS
        ( QUADLIST QUADSET)))
```

```
 EXPR)
(DEFPROP ADDQUADTOLIST
 (LAMBDA (QUAD QUADSET)
  (LIST ( QUADSETNAME QUADSET)
        ( STARTSTATE QUADSET)
        ( REGISTERLIST QUADSET)
        (CONS QUAD ( QUADLIST QUADSET))))
 EXPR)
(DEFPROP DELETEFROMQUADLIST
 (LAMBDA (QUADSET QUADNAME)
  (LIST ( QUADSETNAME QUADSET)
        ( STARTSTATE QUADSET)
        ( REGISTERLIST QUADSET)
        (DELETEFROM QUADSET QUADNAME)))
 EXPR)
(DEFPROP DELETEFROM
 (LAMBDA (SET NAME)
  (COND [(NULL SET) NIL]
        [(EQ NAME ( QUADNAME (CAR SET))) (CDR SET)]
        [(CONS (CAR SET) (DELETEFROM (CDR SET) NAME))]))
 EXPR)
```

A <quad set name> and <start state> are identifiers. <reg> is an infinite list: (*1 *2 *3 ...). A <register list> is any finite initial segment of <reg>, and exhausts all the registers referenced by the <quad set>. It is the responsibility of the user that this condition is satisfied for the <quad set>s he prepares.

A <quad> is similar to a Turing machine quadruple, except that the second and third elements can be any S-expression in LISP. And a <quad name> is added as a fifth, non-operational element. These names are used to identify the quad, and are therefore purely cosmetic.

```
<quad> ::= (<current state> <test>
                <act> <next state> <quad name>)
```

A <current state> and a <next state> are identifiers, pending an upcoming qualification. On the other hand, a <quad name> is a dotted pair:

```
<quad name> ::= ( <quad set name> . <quad name> )
```

The <quad set name> is the name of the <quad set> in which the <quad> occurs, so that all <quad name>s contain the name of the <quad set> in which the <quad> occurs.

```
<test> ::= <S-expression> | ELSE
<act> ::= <S-expression>
<quad list> ::=( <quad>+ )
```

(The sign "+" after a category name indicates any finite non-empty sequence of objects belonging to the category.) Here are the selectors, predicates and modifiers for <quad>s, and the function CHANGEQUADSNAME, which changes the <quad name> for each <quad> in a <quad set>.

A <full state> is like a <full quad name>.

<full state> ::= (<quad set name> . <state>)

A state which is not a <full state> is a <simple state>. The category <state> includes both.

The <next state> of a <quad> may be a <full state> but <current state> is always an atom. It is the user's responsibility to see that this condition is met.

```
(DEFPROP  CURRENTSTATE (LAMBDA (X) (CAR X)) EXPR)
(DEFPROP  TEST (LAMBDA (X) (CADR X)) EXPR)
(DEFPROP  ACT (LAMBDA (X) (CADDR X)) EXPR)
(DEFPROP  NEXTSTATE (LAMBDA (X) (CADDDR X)) EXPR)
(DEFPROP  QUADNAME (LAMBDA (X) (CAR (CDDDDR X))) EXPR)
(DEFPROP FULLQUADNAME
 (LAMBDA (X) (AND [CONSP X] [ATOM (CAR X)] [ATOM (CDR X)]))
 EXPR)
(DEFPROP  SETPARTOFFULLNAME (LAMBDA (X) (CAR X)) EXPR)
(DEFPROP  QUADPARTOFFULLNAME (LAMBDA (X) (CDR X)) EXPR)
(DEFPROP  STATEPARTOFFULLNAME (LAMBDA (X) (CDR X)) EXPR)
(DEFPROP MAKEFULLQUADNAME
 (LAMBDA (SETNAME QUADNAME) (CONS SETNAME QUADNAME))
 EXPR)
```

This next function changes <next state>s in <quad>s to reflect a change in the <quad set name> of the <quad set> containing the <quad>s. It distinguishes <state>s from <full state>s.

```
(DEFPROP UPDATE
 (LAMBDA (QUADS NAME)
  (MAPCAR (FUNCTION (LAMBDA (X) (UPDATENEXTSTATE X NAME)))
          QUADS))
 EXPR)
(DEFPROP UPDATENEXTSTATE
 (LAMBDA (QUAD NAME)
  (LIST ( CURRENTSTATE QUAD)
        ( TEST QUAD)
        ( ACT QUAD)
        (COND [(ATOM ( NEXTSTATE QUAD)) ( NEXTSTATE QUAD)]
              [(CONS NAME
                     ( STATEPARTOFFULLNAME
                      ( NEXTSTATE QUAD)))])
        ( QUADNAME QUAD)))
 EXPR)
```

And here are some service functions for dealing with <quad>s and for searching <quad set>s. FINDELSEQUAD and FINDNONELSEQUADS reconnoitre a list of <quad>s and, for a specified <state> return either a single <quad> whose <current state> is the specified <state> and whose <test> is ELSE, or else a list of non-ELSE <quad>s whose <current state>s are the specified <state>.

```
(DEFPROP CHANGECURRENTSTATE
 (LAMBDA (QUAD STATE)
  (LIST STATE
        ( TEST QUAD)
        ( ACT QUAD)
        ( NEXTSTATE QUAD)
        ( QUADNAME QUAD)))
 EXPR)
(DEFPROP CHANGEQUADTEST
 (LAMBDA (QUAD TEST)
  9
  LIST
  ( CURRENTSTATE QUAD)
  TEST
  ( ACT QUAD)
  ( NEXTSTATE QUAD)
  ( QUADNAME QUAD))
 EXPR)
(DEFPROP CHANGEQUADACT
 (LAMBDA (QUAD ACT)
  (LIST ( CURRENTSTATE QUAD)
        ( TEST QUAD)
        ACT
        ( NEXTSTATE QUAD)
        ( QUADNAME QUAD)))
 EXPR)
(DEFPROP ISELSEQUAD?
 (LAMBDA (QUAD) (EQ ( TEST QUAD) 'ELSE))
 EXPR)
(DEFPROP FINDNONELSEQUADS
 (LAMBDA (QUADSET STATE)
  (COND [(NULL QUADSET) NIL]
        [(AND [EQ ( CURRENTSTATE (CAR QUADSET)) STATE]
              [NOT (ISELSEQUAD? (CAR QUADSET))])
         (CONS (CAR QUADSET)
               (FINDNONELSEQUADS (CDR QUADSET) STATE))]
        [(FINDNONELSEQUADS (CDR QUADSET) STATE)]))
 EXPR)
(DEFPROP FINDELSEQUAD
 (LAMBDA (QUADSET STATE)
  (COND [(NULL QUADSET) NIL]
        [(AND [ISELSEQUAD? (CAR QUADSET)]
              [EQ STATE ( CURRENTSTATE (CAR QUADSET))])
         (CAR QUADSET)]
        [(FINDELSEQUAD (CDR QUADSET) STATE)]))
 EXPR)
(DEFPROP FINDALLQUADS
 (LAMBDA (STATE QUADSET)
  (COND [(NULL QUADSET) NIL]
        [(EQ STATE ( CURRENTSTATE (CAR QUADSET)))
         (CONS (CAR QUADSET)
               (FINDALLQUADS STATE (CDR QUADSET)))]
        [(FINDALLQUADS STATE (CDR QUADSET))]))
 EXPR)
```

Here for example are two typical <quad set>s. The first implements a familiar recursive function to multiply two natural numbers together, and the second implements a process to replace all occurrences of the string "AA" with occurrences of the string "BB".

```
*(SPRINT MUL)
(MUL TA
     (*1 *2 *3)
     ((TA T
          (PROG NIL (SETQ *1 4) (SETQ *2 3) (SETQ *3 0))
          T0
          (MUL . 0))
      (T0 (ZEROP *1) NIL SF (MUL . 1))
      (T0 (ZEROP *2) NIL SF (MUL . 2))
      (T0 ELSE
          (PROG NIL
                (SETQ *3 (*PLUS *2 *3))
                (SETQ *1 (SUB1 *1)))
          T0
          (MUL . 3))))
NIL
*(SPRINT AATOBB)
(AATOBB S0
        (*1 *2)
        ((S0 ELSE
             (PROG NIL
                   (SETQ *1 (CONCAT *1 (LEFT *2)))
                   (SETQ *2 (REST *2)))
             S0
             (AATOBB . 1))
         (S0 (BEGINS "AA" *2)
             (PROG NIL
                   (SETQ *1 (CONCAT *1 "BB"))
                   (SETQ *2 (REST (REST *2))))
             S0
             (AATOBB . 2))))
NIL
```

D. State Transitioning

Control is managed within a <world line> by transitioning from state to state. Within this control process, a <full state>, (<quad set name> . <state>), can direct transition only to a <state> occurring within the <quad set> whose name is <quad set name>. But a <state> can direct transition to any <state> in any <quad set>. In addition to these restrictions, state transitions are limited to <quad set>s whose names appear in a list which is the value of a global identifier, QUADSEARCHLIST. There may also be a list of system <quad>s, FREEQUADS, which the user has access to, and which is always searched unless the flag NO-FREEQUADS is set to T. This flag is accessible by the user. All statements in this document about transitions are understood to acknowledge these rules. Purely as a reminder, we may therefore

speak of "acceptable" <quad>s, "acceptable" <state>s, "acceptable" transitions, and so on.

Suppose the process is in <state> S and finds a <quad>,

(S <test> <act> S' <quad name>)

in a <quad set>,

(<quad set name> <start state> (*1 *2 ... *K) <quad list>)

and that the current values of the registers *1, *2, ..., *K are the S-expressions v1, v2, ..., vK, respectively. Then the following PROG is generated and evaluated:

```
(PROG (*1 *2 ... *K)
      (SETQ *1 'v1)
      (SETQ *2 'v2)
            .
            .
            .
      (SETQ *K 'vK)
      (RETURN
          (COND (<test>
                 <act>
                 (LIST  <quad> (LIST *1 *2 ... *K))))))
```

Given the <values> associated with the <registers>, here is the function which generates the transition program for a specified <quad>, and the function which generates its (varying number of) SETQs.

```
(DEFPROP GENERATEIMMTRANSPROGRAM
 (LAMBDA (VALUES REGISTERS QUAD)
  (EVAL
   (PROG (X)
         (SETQ X
          (APPEND
           (LIST 'PROG REGISTERS)
           (MAKESETQS VALUES REGISTERS)
           (LIST (LIST 'RETURN
                       (LIST 'COND
                             (LIST ( TEST QUAD)
                                   ( ACT QUAD)
                                   (LIST 'LIST
                                         'QUAD
                                         (CONS 'LIST
                                               REGISTERS))))
                       ))))
         (COND [TRANSPRINTFLAG (SPRINT X)])
         (RETURN X))))
 EXPR)
(DEFPROP MAKESETQS
 (LAMBDA (VALUES REGISTERS)
  (COND
```

```
   [(NULL REGISTERS) NIL]
   [(CONS (LIST 'SETQ
                (CAR REGISTERS)
                (COND [(OR [NUMBERP (CAR VALUES)]
                           [STRINGP (CAR VALUES)])
                      (CAR VALUES)]
                     [(LIST 'QUOTE (CAR VALUES))]))
          (MAKESETQS (CDR VALUES) (CDR REGISTERS)))]))
EXPR)
```

The identifier TRANSPRINTFLAG is used to enable the printing of the programs that are generated and EVALuated to decide the question of transition. MAKESETQS terminates on REGISTERS since VALUES comes from another context in the system. In general there may be more values than registers, but there will never be more registers than values.

The transition process is thus implemented so that if the <test> fails, NIL is returned, and if the <test> succeeds, an <item> is returned containing the successful <quad> and the (possibly new) values, v1, v2, ..., vK. GENERATEIMMTRANSPROGRAM generates and evaluates the PROG expression which returns NIL or an <item>.

<item> ::= (<quad> <values>)

If <test> or <act> contains any references to *0 or to the background, these references are not protected. PROGrams like these are <immediate transition program>s.

E. World Lines

A <world line> contains the history of a computation and its current configuration. The <top> of a <world line> contains control information, and its <tail> is a list of <quad>s already successfully processed, from the most recent back to the initial <quad>. The <tail> of a <world line> is followed by its ancestor, and all CDRs of that list are also its ancestors.

```
<world line> ::= (<world line name> <state>
                  <values>
                  <wait flag> <wait condition>
                  <active flag> <tail>
                  <ancestors>)

    <values> ::= <list of S-expressions>
      <wait flag> ::= WAITING | ACTIVE
    <active flag> ::= SUSPENDED | ACTIVE
    <wait condition> ::= <S-expression>
<ancestors> ::= <list of <world line name>s>
```

A <world line>'s <state> is the next state into which it will try to transition. A <world line>'s <values> are its current values for the registers. <ancestors> is a list of the names of the <world line>'s ancestors. Each time a <world line> successfully

transitions to a new state, the resulting <world line> is given a new name, and the <ancestors> of the new <world line> is got by CONSing the old <world line>'s name into the old <world line>'s <ancestors>. A <world line> is said to be suspended if its <active flag> is SUSPENDED, and otherwise it is either waiting or active. A <world line> which is not suspended is waiting if its <wait flag> is WAITING, and otherwise it is active. Here are the selectors and predicates for <world line>s.

```
(DEFPROP  WORLDLINENAME (LAMBDA (X) (CAR X)) EXPR)
(DEFPROP  WORLDLINESTATE (LAMBDA (X) (CADR X)) EXPR)
(DEFPROP  WORLDLINEVALUES (LAMBDA (X) (CADDR X)) EXPR)
(DEFPROP  WAITFLAG (LAMBDA (X) (CADDDR X)) EXPR)
(DEFPROP  WAITCONDITION (LAMBDA (X) (CAR (CDDDDR X))) EXPR)
(DEFPROP  ACTIVEFLAG (LAMBDA (X) (CADR (CDDDDR X))) EXPR)
(DEFPROP  WORLDLINEQUADS
 (LAMBDA (X) (CADDR (CDDDDR X)))
 EXPR)
(DEFPROP  ANCESTORS
 (LAMBDA (X) (CADDDR (CDDDDR X)))
 EXPR)
(DEFPROP ACTIVE?
 (LAMBDA (WLINE)
  (AND [NOT (WAITING? WLINE)] [NOT (SUSPENDED? WLINE)]))
 EXPR)
(DEFPROP SUSPENDED?
 (LAMBDA (WORLDLINE)
  (EQ ( ACTIVEFLAG WORLDLINE) 'SUSPENDED))
 EXPR)
(DEFPROP WAITING?
 (LAMBDA (WORLDLINE)
  (AND [EQ ( WAITFLAG WORLDLINE) 'WAITING]
       [EQ ( ACTIVEFLAG WORLDLINE) 'ACTIVE]))
 EXPR)
```

F. Transition Time

A certain temporal point in the computational process is called transition time, and the transitions from <state> to <state> are completely defined by specifying what happens at transition time. All else is governed by the LISP evaluator, EVAL. First we describe a world line transition, the transition mechanism for one <world line>.

Suppose that an active <world line> has <state> S at transition time. A list of acceptable <quad>s is assembled, each of whose <current state>s is S. All the candidate <quad>s whixch come from one <quad set> are kept together, and if there is an <ELSE quad> it is distinguished.

<candidate quads> ::= (<quad> <quad> ... <ELSE quad>)

That is, an <ELSE quad> appears, if at all, only at the end of this list. If none of the other <quad>s in the <quad list> transitions successfully, then and only then is the <ELSE quad>

tried. The <immediate transition program>s for all these <quad>s are assembled and EVALuated. The result in each case will either be NIL or it will be an <item> containing the <quad> and the register <values> that the <quad>'s processing has produced. All these lists are APPENDed, which makes the failures "go away", so that what remains is a list of items to be taken two at a time. <ELSE quad>s are not processed at all unless all other <quad>s from the same set and with the same <current state> have failed. Here are the functions which find acceptable <quad>s. GETACCEPTABLEQUADS determines if the <world line>'s state is a <simple state> or a <full state>, and conducts the search for quadruples accordingly. It returns as value a <list of<quad list>s>.

<list of <quad list>s> ::= (<quad list> <quad list> ...)

The <state> passed to GETACCEPTABLEQUADS1 is a <simple state> and the value returned is a <list of <quad list>s>. If the global flag, NOFREEQUADS, is T, then the list of FREEQUADS is not searched. The function *APP disposes of NILs, the results of unsuccessful attempts to transition.

```
(DEFPROP GETACCEPTABLEQUADS
 (LAMBDA (STATE SEARCHLIST)
  (COND [(CONSP STATE)
         (FIXCANDIDATES
          (SEARCHQUADSET
           ( STATEPARTOFFULLNAME STATE)
           ( SETPARTOFFULLNAME STATE)))]
        [(GETACCEPTABLEQUADS1 STATE SEARCHLIST)]))
 EXPR)
(DEFPROP GETACCEPTABLEQUADS1
 (LAMBDA (STATE SEARCHLIST)
  (COND [(NULL SEARCHLIST)
         (COND [NOFREEQUADS NIL]
               [(SEARCHQUADSET STATE FREEQUADS)])]
        [(*APPEND (SEARCHQUADSET
                   STATE
                   ( QUADLIST (EVAL (CAR SEARCHLIST))))
                  (GETACCEPTABLEQUADS1
                   STATE
                   (CDR SEARCHLIST)))]))
 EXPR)
```

The <state> passed to SEARCHQUADSET must be a <simple state>.
The function returns a <quad list>
whose last <quad> may, but need not be, an <ELSE quad>.

```
(DEFPROP SEARCHQUADSET
 (LAMBDA (STATE QUADSET)
  (COND [(NULL QUADSET) NIL]
        [(EQ ( STARTSTATE (CAR QUADSET)) STATE)
         (CONS (CAR QUADSET)
               (SEARCHQUADSET STATE (CDR QUADSET)))]
        [(SEARCHQUADSET STATE (CDR QUADSET))]))
 EXPR)
```

After all possible transitions have been attempted, a new <worldline> is constructed for each item returned, by CONSing that item's <quad> to the <tail> of the <world line>, and by installing the values associated with the registers after the <quad> has been processed. If no items are returned, then the <world line> is aborted. Aborted <world line>s "go away" and are irrecoverable. If transitioning fails because <candidate quads> is empty, the <world line> is put on the list DEADWORLDS. The process then returns to transition time. This describes the processing for a single <world line>. Note that the processing of a single <world line> may produce several new <world line>s. In general, the process may encounter, at transition time, a list of <world line>s, so that the process is, from a single <world line> point of view, non-deterministic. All these <world line>s are the elements of a list called CR (Control Register).

At transition time, then, a <world line> transition takes place for each <world line> in CR, and all the resulting <world line>s are entered into a new CR. This process is called CR-transition. Here are the functions which administer CR-transition. TRANSITION1 processes lists of <quad>s and TRANSITION calls TRANSITION1 and distinguishes an <ELSE quad> from other <quad>s. The function TRANSITION returns an <item sequence>.

<item sequence> ::= (<quad> <values> <quad> <values> ...)

where each consecutive <quad> <values> pair is the result of a successful state transition. This is the function that determines whether or not to attempt an <ELSE quad> transition.

```
(DEFPROP TRANSITION
 (LAMBDA (WORLDLINE QUADLIST)
  (COND [(NOT (ACTIVE? WORLDLINE)) NIL]
        [(PROG (A B)
               (SETQ B (ELSEQUADIN QUADLIST))
               (SETQ A
                     (TRANSITION1 WORLDLINE
                                  (NOELSE QUADLIST)))
               (RETURN (COND [(NULL A)
                              (TRANSITION1 WORLDLINE
                                           (LIST B))]
                             [A])))])))
 EXPR)
(DEFPROP TRANSITION1
 (LAMBDA (WORLDLINE QUADLIST)
  (*APP
   (MAPCAR (FUNCTION
            (LAMBDA (X)
             (GENERATEIMMTRANSPROGRAM
              ( WORLDLINEVALUES WORLDLINE)
              ( REGISTERLIST
               (EVAL ( SETPARTOFFULLNAME ( QUADNAME X))))
              X)))
           QUADLIST)))
 EXPR)
```

```
(DEFPROP *APP
 (LAMBDA (X)
  (COND [(NULL X) NIL]
        [(NULL (CAR X)) (*APP (CDR X))]
        [(*APPEND (CAR X) (*APP (CDR X)))]))
 EXPR)

(DEFPROP ELSEQUADIN
 (LAMBDA (X)
  (COND [(NULL X) NIL]
        [(ISELSEQUAD? (CAR X)) (CAR X)]
        [(ELSEQUADIN (CDR X))]))
 EXPR)
(DEFPROP NOELSE
 (LAMBDA (X)
  (COND [(NULL X) NIL]
        [(ISELSEQUAD? (CAR X)) (CDR X)]
        [(CONS (CAR X) (NOELSE (CDR X)))]))
 EXPR)
```

The function MAKENEWWORLDLINES takes an <item sequence> and the <world line> which spawned it as arguments, and generates a list of new <world line>s.

```
(DEFPROP MAKENEWWORLDLINES
 (LAMBDA (ITEMS WORLDLINE)
  (COND [(NULL ITEMS) NIL]
        [(CONS (LIST (INTERN (GENSYM))
                     ( NEXTSTATE (CAR ITEMS))
                     (CADR ITEMS)
                     ( WAITFLAG WORLDLINE)
                     ( WAITCONDITION WORLDLINE)
                     ( ACTIVEFLAG WORLDLINE)
                     (CONS ( QUADNAME (CAR ITEMS))
                           ( WORLDLINEQUADS WORLDLINE))
                     (CONS ( WORLDLINENAME WORLDLINE)
                           ( ANCESTORS WORLDLINE)))
               (MAKENEWWORLDLINES (CDDR ITEMS) WORLDLINE))])
  )
 EXPR)
```

The function ALLNEWWORLDLINES scans a list of active <world line>s and generates the list of new <world line>s spawned by them.

```
(DEFPROP ALLNEWWORLDLINES
 (LAMBDA (WORLDLINES)
  (*APP
   (MAPCAR
    (FUNCTION
     (LAMBDA (X)
      (MAKENEWWORLDLINES
       (COND [(TRANSITION X
                          (GETACCEPTABLEQUADS
                           ( WORLDLINESTATE X)
```

```
                                    QUADSEARCHLIST))]
                        [(SETQ DEADWORLDS (CONS X DEADWORLDS)) NIL])
          X)))
      WORLDLINES)))
 EXPR)
```

After each CR transition, all those <world line>s which are waiting are wait processed. A waiting <world line> is wait processed by EVALuating its <wait condition>. If the value returned is NIL the <world line> is left waiting. Otherwise its <wait flag> is set to ACTIVE.

A full transition is then a two stage process, a sweep of all active <world line>s which are not waiting, and a sweep of all waiting <world line>s.

G. Interrupting

A global identifier, INTERRUPTSTATES, allows the user to interrupt the process. Its value is a list of <state>s or <full state>s. At the completion of each full transition, the <state> in each <world line> is checked for membership in the INTERRUPTSTATES list. CR is searched in CDR order for this purpose. When a <state> in INTERRUPTSTATES occurs, the process is suspended and that <world line>'s name is printed out with a meaningful message. The user may then do any of the things appropriate for user mode. These include suspending a <world line>, reinstating a <world line>, deleting a <world line>, initiating a new <world line>, adding or changing <quad set>s, and so on. Or the user may simply return control to the processor, which will then continue to search for interrupts. INTERRUPTSTATES may be changed by the action of the <quad> transitions. When all interrupt processing is completed, the processor returns to transition time. If the global variable INTERRUPT has the value NIL, then no interrupt processing takes place. If the global variable INTERRUPTWAIT has any value other than NIL, interrupt processing will take place just after the sweep of active <world line>s as well as just after the sweep of all waiting <world lines>. If INTERRUPT is NIL and INTERRUPTWAIT is not NIL, then interrupt processing takes place only just before the sweep of waiting <world line>s. WAITSWEEP examines each waiting world line in CR, and if its <waitcondition> evaluates to NIL it is left waiting. Otherwise its <wait flag> is set to ACTIVE. <wait condition>s are EVALuated in an environment which includes the register values for the <world line>.

The function WAITSWEEP examines each waiting <world line> to see if its <wait condition> EVALuates to T. It appropriately changes the <world line>'s <wait flag>.

```
(DEFPROP WAITSWEEP
 (LAMBDA NIL
  (SETQ CR
   (MAPCAR
```

```
    (FUNCTION
     (LAMBDA (WORLDLINE)
      (COND [(AND [EQ ( ACTIVEFLAG WORLDLINE) 'ACTIVE]
                  [EQ ( WAITFLAG WORLDLINE) 'WAITING]
                  [CAR (GENNNNNNN1* ( WORLDLINEVALUES WORLDL
INE)                                (MAKEWORLDLINEREGISTERS
WORLDLINE)                          (LIST NIL
                                          ( WAITCONDITION WO
RLDLINE)                                NIL
                                          NIL
                                          NIL))]) '
              (ACTIVATE WORLDLINE)]
             [WORLDLINE])))
    CR)))
 EXPR)
(DEFPROP MAKEWORLDLINEREGISTERS
 (LAMBDA (WLINE)
  ((FUNCTION
    (LABEL MK
           (LAMBDA (X REG)
            (COND [(NULL X) NIL]
                  [(CONS (CAR REG) (MK (CDR X) (CDR REG)))])
            )))
   ( WORLDLINEVALUES WLINE)
   REG))
 EXPR)
(DEFPROP ACTIVATE
 (LAMBDA (WORLDLINE)
  (LIST ( WORLDLINENAME WORLDLINE)
        ( WORLDLINESTATE WORLDLINE)
        ( WORLDLINEVALUES WORLDLINE)
        'ACTIVE
        ( WAITCONDITION WORLDLINE)
        ( ACTIVEFLAG WORLDLINE)
        ( WORLDLINEQUADS WORLDLINE)))
 EXPR)
```

And here is KOTSU's top level, with its important supporting function, ONANDON.

```
(DEFPROP ONANDON (LAMBDA NIL NEXTSTEP) EXPR)

(DEFPROP GOKOTSU
 (LAMBDA NIL
  (PROG  (X SUSPENDED WAITING ACTIVE INTERRUPT
            INTERRUPTSTATES INTERRUPTWAIT NEXTSTEP
            QUADSEARCHLIST FREEQUADS NOFREEQUADS STUFF ELSE
            TRANSPRINTFLAG INTERRUPTED REG)
         (PRINC "KOTSU GETTING STARTED!")
         (SETQ ELSE T)
         (SETQ REG
               '(*1 *2 *3 *4 *5 *6 *7 *8 *9 *10 *11 *12 *13
                    *14 *15 *16))
         (TERPRI)
 INITIALIZE
```

```
        (SETIT 'DEBUGMODE)
        (SETIT 'DEADWORLDS)
        (SETIT 'QUADSEARCHLIST)
        (SETIT 'NOFREEQUADS)
        (SETIT 'CR)
        (SETIT 'FREEQUADS)
        (SETIT 'TRANSPRINTFLAG)
        (SETIT 'INTERRUPT)
        (SETIT 'INTERRUPTSTATES)
        (SETIT 'INTERRUPTWAIT)
        (SETQ NEXTSTEP 'WORLDLINESWEEPING)
        (GO INTERRUPTING)
WORLDLINESWEEPING
        (NEXT)
        (COND [(NOT INTERRUPT) (GO WAITSWEEPING)])
        (SETQ NEXTSTEP 'WAITSWEEPING)
        (GO INTERRUPTING)
WAITSWEEPING
        (WAITSWEEP)
        (COND [(NOT INTERRUPTWAIT) (GO WORLDLINESWEEPING)])
        (SETQ NEXTSTEP 'WORLDLINESWEEPING)
        (GO INTERRUPTING)
INTERRUPTING
        (COND
         [DEBUGMODE (GO INTERRUPTING1)]
         [(NULL
           (SETQ INTERRUPTED
            (*APPSIES
             (MAPCAR
              (FUNCTION
               (LAMBDA (WLINE)
                (COND [(MEMQ ( WORLDLINESTATE WLINE)
                             INTERRUPTSTATES)
                       WLINE]
                      [NIL])))
              CR))))
          (GO (ONANDON))])
INTERRUPTING1
        (TERPRI)
        (PRINC "INTERRUPT MODE!")
        (TERPRI)
        (PRINC "WHAT NOW:")
        (SETQ X (READ))
        (COND [(EQ X 'CONTINUE) (GO CONTINUE)]
              [(EQ X 'STOP) (RETURN 'MU!)]
              [(EQ X 'RESTART) (GO RESTART)]
              [(EQ X 'EV) (GO EV)]
              [(EQ X 'HOW) (SHOWHOWITIS)]
              [(PRINC "TYPE ONE OF: CONTINUE, STOP, RESTART,
EV, HOW.")     (TERPRI)
               (GO INTERRUPTING1)])
        (GO INTERRUPTING1)
CONTINUE
        (PRINC "TRANSITIONING AGAIN.")
        (TERPRI)
```

```
        (GO (ONANDON))
RESTART
        (GO INITIALIZE)
    EV (PRINC "EV WHAT?")
        (TERPRI)
        (SPRINT (EVAL (READ)))
        (TERPRI)
        (GO INTERRUPTING)))
 EXPR)
(DEFPROP NEXT
 (LAMBDA NIL (SPRINT (SETQ CR (ALLNEWWORLDLINES CR))))
 EXPR)
```

The handling of interrupted <world line>s has not been checked out and is not included here. This process will be installed just before the label INTERRUPTING1.

SETIT initializes a global variable by adding a ":" to its name for a prompt symbol, and then by SETting that variable to the result of (READ). SHOWHOWITIS displays the values of all these global variables.

```
(DEFPROP SETIT
 (LAMBDA (X)
  (PROG NIL (PRINC (CONCAT X ":")) (SET X (READ))))
 EXPR)
(DEFPROP SHOWHOWITIS
 (LAMBDA NIL
  (PROG NIL
        (MAPCAR (FUNCTION SHOWOUT)
                '(DEADWORLDS QUADSEARCHLIST NOFREEQUADS
                             FREEQUADS TRANSPRINTFLAG
                             INTERRUPTSTATES INTERRUPTWAIT
                             ELSE))
        (PRINC "CR:")
        (SPRINT CR 12)))
 EXPR)
(DEFPROP SHOWOUT
 (LAMBDA (X)
  (PROG NIL
        (PRINC (CONCAT X ":"))
        (SPRINT (EVAL X) 8)
        (TERPRI)))
 EXPR)
```

DISPLAY, DISPLAY1 and SHORTEN form a set of functions to display a <world line> in an organized format on the terminal.

```
(DEFPROP DISPLAY
 (LAMBDA (NAME)
  (MAPCAR (FUNCTION (LAMBDA (X) (DISPLAY1 NAME X))) CR))
 EXPR)
(DEFPROP DISPLAY1
 (LAMBDA (NAME X)
```

```
  (SPRINT (COND [(EQ NAME ( WORLDLINENAME X)) (SHORTEN X)]))
  )
 EXPR)
(DEFPROP SHORTEN
 (LAMBDA (X)
  (LIST ( WORLDLINENAME X)
        ( WORLDLINESTATE X)
        ( WORLDLINEVALUES X)
        ( WAITFLAG X)
        ( WAITCONDITION X)
        ( ACTIVEFLAG X)
        (CAR ( WORLDLINEQUADS X))
        ( WORLDLINENAMES X)))
 EXPR)
```

H. Status Changes

An important family of facilities in this system allows the modification of the status of particular <world line>s in the universe, where the status of a <world line> is determined by whether that <world line> is waiting, active or suspended. In addition, there is a DEADWORLDS register which comes to contain all those <world line>s which were active at transition time but were unable to complete a transition to a new <state>. There are facilities to move a <world line> arbitrarily from DEADWORLDS to CR or from CR to DEADWORLDS. It will be useful to make such a move if other changes have also occurred elsewhere in the universe. The user may initiate these moves at interrupt time, or they may occur under program control.

A particular <world line> may be identified by name, but since the computations allowed may be non-deterministic that name may come to be associated with a number of distinct descendents of the original <world line>. To accommodate these realities we have the notion of a <tribe>. A <tribe> is expressed as a <world line name> and is the set of all those <world line>s whose <ancestors> contain that <world line name>.

The decision to move a <world line> between CR and DEADWORLDS can in general be made contingent upon the outcome of EVALuating a specified expression called a <condition>. A <condition> is not met if it EVALuates to NIL and is met if it EVALuates to anything else. Thus, in general, movement of <world line>s only takes place if a specified <condition> is met. One can further restrict the movement of <world line>s between CR and DEAD. Among those <world line>s belonging to a specified <tribe> one can specify that only those which are active are to be moved, or that only those which are suspended are to be moved, or only those which are waiting.

In a related facility a <world line> may be left on CR, but its status may be changed. One can command that all those active <world line>s belonging to a specified <tribe> are to be made waiting, or are to be made suspended. Or else that all "tribal" <world line>s that are waiting are to be made active. And these

changes can also be made contingent upon a certain <condition>'s being met.

Finally, all these changes to a <world line>'s status can be implicated with the provisional character of changes to the values of the registers, *1, *2, ... A specified <condition> of sufficient complexity may make changes to the values of certain registers, and it is an option to determine whether or not these register values are to be protected in case the <condition> is not met.

All these realities are concretized in a function that takes four arguments.

(MAYIFYWORLDLINES TRIBE CONDITION ACTION PROTECTION)

The argument TRIBE is the <tribe> name. CONDITION is any LISP S-expression. It is to be EVALuated once for each tribal <world line> in WORLDLINES and in the presence of the binding of that <world line>'s values to the registers. PROTECTION is T or NIL, where T indicates that register values are to be protected if the <condition> fails, and NIL indicates that changes to these registers are to be made a permanent part of the <world line> whether or not the <condition> is met. WORLDLINE is normally CR but it may be DEADWORLDS or any other list of <world line>s for that matter.

ACTION specifies further restrictions to the status of a <world line> in WORLDLINES. Action is normally the name of a function which accepts a <world line> as its argument and gives back as value that <world line> with its status changed.

We turn now to a detailed review of the LISP functions which implement these status changes.

The following eight functions take a <world line> as argument and give as value that <world line> with its status appropriately changed, but only subject to a certain kind of restriction. SUSPEND! only suspends an active <world line>, and DESUSPEND!activates only a suspended <world line>. WAITPLEASE! sets only an active <world line> waiting, and UNWAIT! activates only waiting <world line>s. On the other hand, SUSPENDALLCASES!, DESUSPENDALLCASES!, WAITPLEASEALLCASES! and UNWAITALLCASES! install the specified changes in a <world line> regardless of its original status. The function NEWVALUES inserts a new list of <values> into a <world line>.

```
(DEFPROP SUSPEND!
 (LAMBDA (WORLDLINE)
  (COND [(ACTIVE? WLINE)
         (LIST ( WORLDLINENAME WORLDLINE)
               ( WORLDLINESTATE WORLDLINE)
               ( WORLDLINEVALUES WORLDLINE)
               ( WAITFLAG WORLDLINE)
               ( WAITCONDITION WORLDLINE)
```

```
                'SUSPENDED
                ( WORLDLINEQUADS WORLDLINE)
                ( ANCESTORS WORLDLINE))]
          [WLINE]))
 EXPR)
(DEFPROP DESUSPEND!
 (LAMBDA (WORLDLINE)
  (COND [(SUSPENDED! WLINE)
          (LIST ( WORLDLINENAME WORLDLINE)
                ( WORLDLINESTATE WORLDLINE)
                ( WORLDLINEVALUES WORLDLINE)
                ( WAITFLAG WORLDLINE)
                ( WAITCONDITION WORLDLINE)
                'ACTIVE
                ( WORLDLINEQUADS WORLDLINE)
                ( ANCESTORS WORLDLINE))]
          [WLINE]))
 EXPR)
(DEFPROP WAITPLEASE!
 (LAMBDA (WORLDLINE)
  (COND [(ACTIVE? WLINE)
          (LIST ( WORLDLINENAME WORLDLINE)
                ( WORLDLINESTATE WORLDLINE)
                ( WORLDLINEVALUES WORLDLINE)
                'WAITING
                ( WAITCONDITION WORLDLINE)
                ( ACTIVEFLAG WORLDLINE)
                ( WORLDLINEQUADS WORLDLINE)
                ( ANCESTORS WORLDLINE))]
          [WLINE]))
 EXPR)
(DEFPROP UNWAIT!
 (LAMBDA (WORLDLINE)
  (COND [(WAITING? WLINE)
          (LIST ( WORLDLINENAME WORLDLINE)
                ( WORLDLINESTATE WORLDLINE)
                ( WORLDLINEVALUES WORLDLINE)
                'ACTIVE
                ( WAITCONDITION WORLDLINE)
                ( ACTIVEFLAG WORLDLINE)
                ( WORLDLINEQUADS WORLDLINE)
                ( ANCESTORS WORLDLINE))]
          [WLINE]))
 EXPR)
(DEFPROP SUSPENDALLCASES!
 (LAMBDA (WORLDLINE)
  (LIST ( WORLDLINENAME WORLDLINE)
        ( WORLDLINESTATE WORLDLINE)
        ( WORLDLINEVALUES WORLDLINE)
        ( WAITFLAG WORLDLINE)
        ( WAITCONDITION WORLDLINE)
        'SUSPENDED
        ( WORLDLINEQUADS WORLDLINE)
        ( ANCESTORS WORLDLINE)))
 EXPR)
```

```
(DEFPROP DESUSPENDALLCASES!
 (LAMBDA (WORLDLINE)
  (LIST ( WORLDLINENAME WORLDLINE)
        ( WORLDLINESTATE WORLDLINE)
        ( WORLDLINEVALUES WORLDLINE)
        ( WAITFLAG WORLDLINE)
        ( WAITCONDITION WORLDLINE)
        'ACTIVE
        ( WORLDLINEQUADS WORLDLINE)
        ( ANCESTORS WORLDLINE)))
 EXPR)
(DEFPROP WAITPLEASEALLCASES!
 (LAMBDA (WORLDLINE)
  (LIST ( WORLDLINENAME WORLDLINE)
        ( WORLDLINESTATE WORLDLINE)
        ( WORLDLINEVALUES WORLDLINE)
        'WAITING
        ( WAITCONDITION WORLDLINE)
        ( ACTIVEFLAG WORLDLINE)
        ( WORLDLINEQUADS WORLDLINE)
        ( ANCESTORS WORLDLINE)))
 EXPR)
(DEFPROP UNWAITALLCASES!
 (LAMBDA (WORLDLINE)
  (LIST ( WORLDLINENAME WORLDLINE)
        ( WORLDLINESTATE WORLDLINE)
        ( WORLDLINEVALUES WORLDLINE)
        'ACTIVE
        ( WAITCONDITION WORLDLINE)
        ( ACTIVEFLAG WORLDLINE)
        ( WORLDLINEQUADS WORLDLINE)
        ( ANCESTORS WORLDLINE)))
 EXPR)
(DEFPROP NEWVALUES
 (LAMBDA (WORLDLINE)
  (LIST ( WORLDLINENAME WORLDLINE)
        ( WORLDLINESTATE WORLDLINE)
        VALUES
        ( WAITFLAG WORLDLINE)
        ( WAITCONDITION WORLDLINE)
        ( ACTIVEFLAG WORLDLINE)
        ( WORLDLINEQUADS WORLDLINE)
        ( ANCESTORS WORLDLINE)))
 EXPR)
(DEFPROP GENNNNNNN1*
 (LAMBDA (VALUES REGISTERS QUAD)
  (EVAL
   (PROG (RESULT)
         (SETQ RESULT
               (APPEND (LIST 'PROG REGISTERS)
                       (MAKESETQS VALUES REGISTERS)
                       (LIST (LIST 'RETURN
                                   (LIST 'LIST
                                         ( TEST QUAD)
                                         (CONS 'LIST
```

```
                                                        REGISTERS))))
                ))
          (COND [TRANSPRINTFLAG (SPRINT RESULT)])
          (RETURN RESULT))))
EXPR)
```

The function GENNNNNNN1* yields a list of two things - the result of EVALuating a <condition> in an environment in which the <world line>'s values are bound to the registers, and the possibly changed values themselves. We might call this a <condition item>.

<condition item> ::= (<S-expression value> <values>)

Register protection is indicated by passing a non-NIL value for PROTECTION to the function MAYIFYWORLDLINES.

```
(DEFPROP MAYIFYWORLDLINES
 (LAMBDA (TRIBE CONDITION ACTION PROTECTION)
  (COND
   [(TRANSMIGRATE? ACTION)
    ((CAR ACTION) TRIBE CONDITION (CDR ACTION) PROTECTION)]
   [(SETQ CR
     (MAPCAR
      (FUNCTION
       (LAMBDA (WLINE)
         (COND
          [(INTRIBE? WLINE TRIBE)
           (PROG (RESULT)
                 (SETQ RESULT
                       (GENNNNNNN1* ( WORLDLINEVALUES WLINE)
                                    (MAKEWORLDLINEREGISTERS W
LINE)                               (LIST NIL CONDITION NIL
                                          NIL NIL)))
                 (RETURN (COND [(CAR RESULT)
                                (NEWVALUES (CADR RESULT)
                                           (ACTION WLINE))]
                               [PROTECTION WLINE]
                               [(NEWVALUES (CADR RESULT)
                                           WLINE)])))]
          [WLINE])))
      CR))]))
 EXPR)
(DEFPROP TRANSMIGRATE? (LAMBDA (X) (CONSP X)) EXPR)
```

The outer condition determines whether the status change consists in a move between CR and DEADWORLDS. If so, the result is determined by the function TRANSMIGRATION. Otherwise a specified function is mapped to each <world line> in WORLDLINES, assembling a new list of <world line>s. This inner function applies only to <world line>s belonging to the tribe. Given that distinction, it has three cases. The first condition holds when the <condition> EVALuates to non-NIL in the presence of the <world line>'s register <values>, and it returns the <world line>

with its <values> reflecting possible changes to its <values> as well as with its status appropriately modified. If CONDITION has EVALuated to NIL the second case sees if the registers are to be protectud. If so, it returns the <world line> unchanged. In the third case, where the registers are not to be protected, the possibly new register values are installed.

Status changes are themselves incorporated by applying the argument ACTION to the <world line>. ACTION must therefore be either the name of a function or else a lambda expression, in either case denoting a function with one argument.

The part of the universe that is stored in CR is what is changed by a call on MAYIFYWORLDLINES. The function MAKEWORLD-LINEREGISTERS uses the <world line>'s <values> list to construct a list of registers. The function INTRIBE? asks if the specified <world line> belongs to the <tribe>. That is, does the <tribe> name belong the <world line>'s <ancestors>?

```
(DEFPROP MAKEWORLDLINEREGISTERS
 (LAMBDA (WLINE)
  ((FUNCTION
    (LABEL MK
           (LAMBDA (X REG)
            (COND [(NULL X) NIL]
                  [(CONS (CAR REG) (MK (CDR X) (CDR REG)))])
            )))
   ( WORLDLINEVALUES WLINE)
   REG))
 EXPR)
(DEFPROP INTRIBE?
 (LAMBDA (WORLDLINE TRIBE)
  (MEMQ TRIBE ( ANCESTORS WORLDLINE)))
 EXPR)
```

TRANSMIGRATE determines whether the action involves movement between CR and DEAD. The third argument to MAYIFYWORLDLINES will in these cases be a dotted pair whose CAR is one of TRANSCRTO-DEAD/TRANSDEADTOCR, and whose CDR is one of ACTIVE?/SUSPENDED?/WAITING?/ALL. This status is used to restrict <world line>s for transfer. The function ALL? is the constant function whose argument is a <world line> and whose value is always T. TRANSCRTODEAD and TRANSDEADTOCR are PROGs, and they are exactly alike except for an interchange of the names CR and DEAD, and their local cognates. The COND after NEXT selects only tribal <world line>s having the appropriate status. The SETQ after ONN generates the result of EVALuating the <condition> in the presence of register bindings. The final COND determines whether to move the <world line> and how to dispose of changes to the register <values> for that <world line>.

```
(DEFPROP TRANSCRTODEAD
 (LAMBDA (TRIBE CONDITION STATUS PROTECTION)
```

```
 (PROG (LOCALDEAD RESULT IT)
       (SETQ LOCALCR CR)
  NEXT (COND [(NULL LOCALCR) (RETURN NIL)]
             [(AND [INTRIBE? (SETQ IT (CAR LOCALCR)) TRIBE]
                   [STATUS IT])
              (GO ONN)]
             [(SETQ CR (CONS IT CR))
              (SETQ LOCALCR (CDR LOCALCR))
              (GO NEXT)])
   ONN (SETQ RESULT
             (GENNNNNNN1* ( WORLDLINEVALUES IT)
                          (MAKEWORLDLINEREGISTERS IT)
                          (LIST NIL CONDITION NIL NIL NIL))
       )
       (COND [(CAR RESULT)
              (SETQ DEAD
                    (CONS (NEWVALUES (CADR RESULT) IT)
                          DEAD))
              (SETQ LOCALCR (CDR LOCALCR))
              (GO NEXT)]
             [PROTECTION (SETQ CR (CONS IT CR))
                         (SETQ LOCALCR (CDR LOCALCR))]
             [(SETQ CR
                    (CONS (NEWVALUES (CADR RESULT) IT) CR))
              (GO NEXT)])))
EXPR)
(DEFPROP TRANSDEADTOCR
 (LAMBDA (TRIBE CONDITION STATUS PROTECTION)
  (PROG (LOCALDEAD RESULT IT)
        (SETQ LOCALDEAD DEAD)
  NEXT (COND [(NULL LOCALDEAD) (RETURN NIL)]
             [(AND [INTRIBE? (SETQ IT (CAR LOCALDEAD))
                             TRIBE]
                   [STATUS IT])
              (GO ONN)]
             [(SETQ DEAD (CONS IT DEAD))
              (SETQ LOCALDEAD (CDR LOCALDEAD))
              (GO NEXT)])
   ONN (SETQ RESULT
             (GENNNNNNN1* ( WORLDLINEVALUES IT)
                          (MAKEWORLDLINEREGISTERS IT)
                          (LIST NIL CONDITION NIL NIL NIL))
       )
       (COND [(CAR RESULT)
              (SETQ CR
                    (CONS (NEWVALUES (CADR RESULT) IT) CR))
              (SETQ LOCALDEAD (CDR LOCALDEAD))
              (GO NEXT)]
             [PROTECTION (SETQ DEAD (CONS IT DEAD))
                         (SETQ LOCALDEAD (CDR LOCALDEAD))]
             [(SETQ DEAD
                    (CONS (NEWVALUES (CADR RESULT) IT)
                          DEAD))
              (GO NEXT)])))
EXPR)
```

The functions TRANSCRTODEAD and TRANSDEADTOCR have not been tested, but are included here for completeness. The other processes for status change have been applied successfully to non-trivial test cases.

The functions MAKE and NAMETHOSE are used to make it easy to enter quad sets into the system.

```
(DEFPROP MAKE
 (LAMBDA NIL
  (PROG (NAME QUADS STATE REG X)
        (PRINC "QUAD SET NAME:")
        (SETQ NAME (READ))
        (SETQ QUADS NIL)
   NEXT (PRINC "NEXT QUAD:")
        (COND [(NULL (SETQ X (READ))) (GO ON)])
        (SETQ QUADS (CONS (FIXACTINQUAD X) QUADS))
        (GO NEXT)
     ON (PRINC "START STATE:")
        (SETQ STATE (READ))
        (PRINC "LIST OF ALL REGISTERS:")
        (SETQ REG (READ))
        (SET NAME
             (LIST NAME STATE REG (NAMETHOSE QUADS NAME 1)))
  ))
 EXPR)
(DEFPROP NAMETHOSE
 (LAMBDA (X NAME N)
  (COND [(NULL X) NIL]
        [(CONS (NCONC1 (CAR X) (CONS NAME N))
               (NAMETHOSE (CDR X) NAME (ADD1 N)))]))
 EXPR)
(DEFPROP *APPSIES
 (LAMBDA (X)
  (COND [(NULL X) NIL]
        [(NULL (CAR X)) (*APPSIES (CDR X))]
        [(CONS (CAR X) (*APPSIES (CDR X)))]))
 EXPR)
```

I. Simple Test Case

We include here a record of a brief sample terminal session. KOTSU's awkwardness and primitivity are apparent at this stage. This test shows that the transition mechanism functions as advertised, and that the status change mechanisms are also functional for the cases shown. Time considerations prevent our including a significant application of these new facilities.

```
R LISP 60
ALLOC? (Y OR N) N
Rutgers/UCI LISP - 06/02/78 [Syracuse University 08/14/80]
*(CSYM G0010)
G0010
*(LINELENGTH 60)
```

```
60
*(DSKIN KOTSU)(DSKIN KOTSU1)
***
***
*(GOKOTSU)
KOTSU GETTING STARTED!
DEBUGMODE:*T
DEADWORLDS:*NIL
QUADSEARCHLIST:*(AATOBB BBTOCC CCTODD)
NOFREEQUADS:*T
CR:*NIL
FREEQUADS:*T
TRANSPRINTFLAG:*T
INTERRUPT:*T
INTERRUPTSTATES:*NIL
INTERRUPTWAIT:*NIL
INTERRUPT MODE!
WHAT NOW:*EV
EV WHAT?
*(SETQ CR CR1)
((G0005 S0 ("" "AABBCCDD") ACTIVE T ACTIVE NIL NIL))
INTERRUPT MODE!
WHAT NOW:*CONTINUE
TRANSITIONING AGAIN.
(PROG (*1 *2)
      (SETQ *1 "")
      (SETQ *2 "AABBCCDD")
      (RETURN (COND [(BEGINS "AA" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "BB"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "")
      (SETQ *2 "AABBCCDD")
      (RETURN (COND [(BEGINS "BB" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "CC"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "")
      (SETQ *2 "AABBCCDD")
      (RETURN (COND [ELSE (PROG NIL
                                (SETQ *1
                                      (CONCAT *1 (LEFT *2)))
                                (SETQ *2 (REST *2)))
                          (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "")
      (SETQ *2 "AABBCCDD")
      (RETURN (COND [(BEGINS "CC" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "DD"))
                           (SETQ *2 (REST (REST *2))))
```

```
                        (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "")
      (SETQ *2 "AABBCCDD")
      (RETURN (COND [ELSE (PROG NIL
                                (SETQ *1
                                      (CONCAT *1 (LEFT *2)))
                                (SETQ *2 (REST *2)))
                          (LIST QUAD (LIST *1 *2))]))))
((G0013 S0
        ("BB" "BBCCDD")
        ACTIVE
        T
        ACTIVE
        ((AATOBB . 2))
        (G0005))
 (G0014 S0
        ("A" "ABBCCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1))
        (G0005))
 (G0015 S0
        ("A" "ABBCCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1))
        (G0005)))
INTERRUPT MODE!
WHAT NOW:*CONTINUE
TRANSITIONING AGAIN.
(PROG (*1 *2)
      (SETQ *1 "BB")
      (SETQ *2 "BBCCDD")
      (RETURN (COND [(BEGINS "AA" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "BB"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "BB")
      (SETQ *2 "BBCCDD")
      (RETURN (COND [(BEGINS "BB" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "CC"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "BB")
      (SETQ *2 "BBCCDD")
      (RETURN (COND [ELSE (PROG NIL
                                (SETQ *1
                                      (CONCAT *1 (LEFT *2)))
```

```
                                    (SETQ *2 (REST *2)))
                                (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "BB")
      (SETQ *2 "BBCCDD")
      (RETURN (COND [(BEGINS "CC" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "DD"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "BB")
      (SETQ *2 "BBCCDD")
      (RETURN (COND [ELSE (PROG NIL
                                (SETQ *1
                                      (CONCAT *1 (LEFT *2)))
                                (SETQ *2 (REST *2)))
                          (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [(BEGINS "AA" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "BB"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [(BEGINS "BB" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "CC"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [ELSE (PROG NIL
                                (SETQ *1
                                      (CONCAT *1 (LEFT *2)))
                                (SETQ *2 (REST *2)))
                          (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [(BEGINS "CC" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "DD"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [ELSE (PROG NIL
                                (SETQ *1
```

```
                                        (CONCAT *1 (LEFT *2)))
                                    (SETQ *2 (REST *2)))
                                (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [(BEGINS "AA" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "BB"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [(BEGINS "BB" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "CC"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [ELSE (PROG NIL
                                (SETQ *1
                                      (CONCAT *1 (LEFT *2)))
                                (SETQ *2 (REST *2)))
                          (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [(BEGINS "CC" *2)
                     (PROG NIL
                           (SETQ *1 (CONCAT *1 "DD"))
                           (SETQ *2 (REST (REST *2))))
                     (LIST QUAD (LIST *1 *2))])))
(PROG (*1 *2)
      (SETQ *1 "A")
      (SETQ *2 "ABBCCDD")
      (RETURN (COND [ELSE (PROG NIL
                                (SETQ *1
                                      (CONCAT *1 (LEFT *2)))
                                (SETQ *2 (REST *2)))
                          (LIST QUAD (LIST *1 *2))])))
((G0016 S0
        ("BBCC" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 2) (AATOBB . 2))
        (G0013 G0005))
 (G0017 S0
        ("BBB" "BCCDD")
        ACTIVE
        T
        ACTIVE
```

```
          ((BBTOCC . 1) (AATOBB . 2))
          (G0013 G0005))
 (G0018 S0
          ("BBB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (AATOBB . 2))
          (G0013 G0005))
 (G0019 S0
          ("AA" "BBCCDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (BBTOCC . 1))
          (G0014 G0005))
 (G0020 S0
          ("AA" "BBCCDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (BBTOCC . 1))
          (G0014 G0005))
 (G0021 S0
          ("AA" "BBCCDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (CCTODD . 1))
          (G0015 G0005))
 (G0022 S0
          ("AA" "BBCCDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (CCTODD . 1))
          (G0015 G0005)))
INTERRUPT MODE!
WHAT NOW:*EV
EV WHAT?
*(SETQ TRANSPRINTFLAG NIL)
NIL
INTERRUPT MODE!
WHAT NOW:*CONTINUE
TRANSITIONING AGAIN.
((G0023 S0
          ("BBCCC" "CDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (BBTOCC . 2) (AATOBB . 2))
          (G0016 G0013 G0005))
 (G0024 S0
          ("BBCCDD" "DD")
          ACTIVE
```

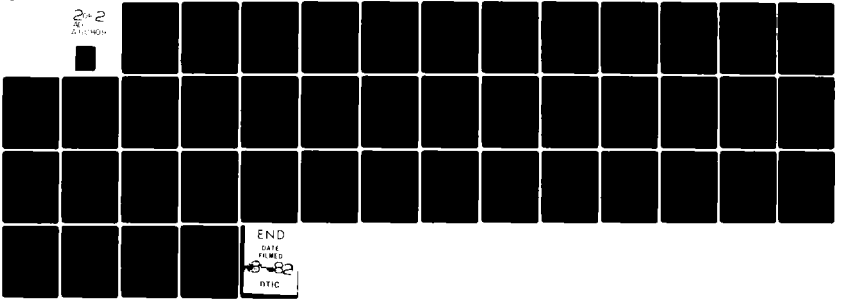
```
        T
        ACTIVE
        ((CCTODD . 2) (BBTOCC . 2) (AATOBB . 2))
        (G0016 G0013 G0005))
(G0025 S0
        ("BBCCC" "CDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (BBTOCC . 2) (AATOBB . 2))
        (G0016 G0013 G0005))
(G0026 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (BBTOCC . 1) (AATOBB . 2))
        (G0017 G0013 G0005))
(G0027 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (BBTOCC . 1) (AATOBB . 2))
        (G0017 G0013 G0005))
(G0028 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (CCTODD . 1) (AATOBB . 2))
        (G0018 G0013 G0005))
(G0029 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (CCTODD . 1) (AATOBB . 2))
        (G0018 G0013 G0005))
(G0030 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 2) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0031 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0032 S0
        ("AAB" "BCCDD")
```

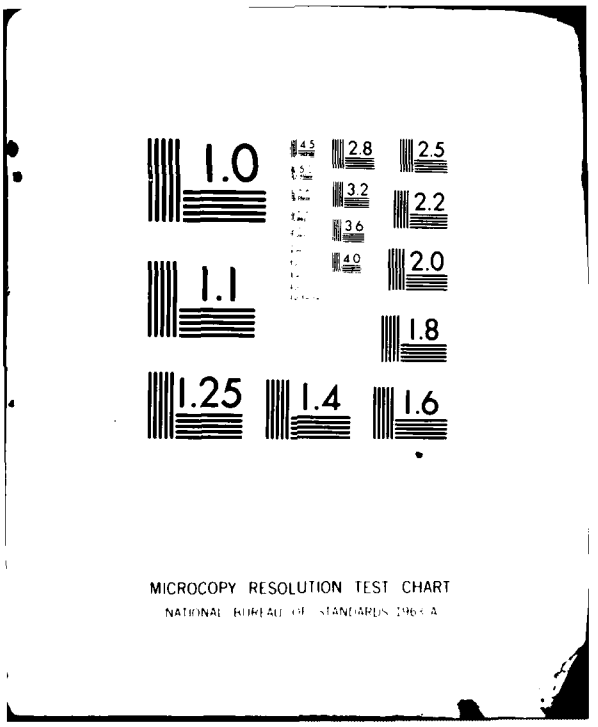
```
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0033 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 2) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0034 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0035 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0036 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 2) (BBTOCC . 1) (CCTODD . 1))
        (G0021 G0015 G0005))
(G0037 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (BBTOCC . 1) (CCTODD . 1))
        (G0021 G0015 G0005))
(G0038 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (BBTOCC . 1) (CCTODD . 1))
        (G0021 G0015 G0005))
(G0039 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 2) (CCTODD . 1) (CCTODD . 1))
        (G0022 G0015 G0005))
(G0040 S0
```

```
          ("AAB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (CCTODD . 1) (CCTODD . 1))
          (G0022 G0015 G0005))
 (G0041 S0
          ("AAB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (CCTODD . 1) (CCTODD . 1))
          (G0022 G0015 G0005)))
INTERRUPT MODE!
WHAT NOW:*EV
EV WHAT?
*(MAYIFYWORLDLINES 'G0014 T 'SUSPEND! T)
((G0023 S0
          ("BBCCC" "CDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (BBTOCC . 2) (AATOBB . 2))
          (G0016 G0013 G0005))
 (G0024 S0
          ("BBCCDD" "DD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 2) (BBTOCC . 2) (AATOBB . 2))
          (G0016 G0013 G0005))
 (G0025 S0
          ("BBCCC" "CDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (BBTOCC . 2) (AATOBB . 2))
          (G0016 G0013 G0005))
 (G0026 S0
          ("BBBB" "CCDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (BBTOCC . 1) (AATOBB . 2))
          (G0017 G0013 G0005))
 (G0027 S0
          ("BBBB" "CCDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (BBTOCC . 1) (AATOBB . 2))
          (G0017 G0013 G0005))
 (G0028 S0
          ("BBBB" "CCDD")
          ACTIVE
```

AD-A110 809 SYRACUSE UNIV NY SCHOOL OF COMPUTER AND INFORMATION --ETC F/G 9/2
INTEGRATED PARALLEL PROCESSES: THE ELEMENTS OF MEANING IN LANGU--ETC(U)
NOV 81 E F STORM F30602-77-C-0235

UNCLASSIFIED RADC-TR-80-379-VOL-4 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

```
        T
        ACTIVE
        ((BBTOCC . 1) (CCTODD . 1) (AATOBB . 2))
        (G0018 G0013 G0005))
(G0029 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (CCTODD . 1) (AATOBB . 2))
        (G0018 G0013 G0005))
(G0030 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        SUSPENDED
        ((BBTOCC . 2) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0031 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        SUSPENDED
        ((BBTOCC . 1) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0032 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        SUSPENDED
        ((CCTODD . 1) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0033 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        SUSPENDED
        ((BBTOCC . 2) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0034 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        SUSPENDED
        ((BBTOCC . 1) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0035 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        SUSPENDED
        ((CCTODD . 1) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0036 S0
        ("AACC" "CCDD")
```

```
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 2) (BBTOCC . 1) (CCTODD . 1))
          (G0021 G0015 G0005))
 (G0037 S0
          ("AAB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (BBTOCC . 1) (CCTODD . 1))
          (G0021 G0015 G0005))
 (G0038 S0
          ("AAB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (BBTOCC . 1) (CCTODD . 1))
          (G0021 G0015 G0005))
 (G0039 S0
          ("AACC" "CCDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 2) (CCTODD . 1) (CCTODD . 1))
          (G0022 G0015 G0005))
 (G0040 S0
          ("AAB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (CCTODD . 1) (CCTODD . 1))
          (G0022 G0015 G0005))
 (G0041 S0
          ("AAB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (CCTODD . 1) (CCTODD . 1))
          (G0022 G0015 G0005)))
NIL
INTERRUPT MODE!
WHAT NOW:*EV
EV WHAT?
*(MAYIFYWORLDLINES 'G0014  '(BEGINS "BC" *2) 'DESUSPEND! T)
((G0023 S0
          ("BBCCC" "CDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (BBTOCC . 2) (AATOBB . 2))
          (G0016 G0013 G0005))
 (G0024 S0
          ("BBCCDD" "DD")
          ACTIVE
```

```
        T
        ACTIVE
        ((CCTODD . 2) (BBTOCC . 2) (AATOBB . 2))
        (G0016 G0013 G0005))
(G0025 S0
        ("BBCCC" "CDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (BBTOCC . 2) (AATOBB . 2))
        (G0016 G0013 G0005))
(G0026 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (BBTOCC . 1) (AATOBB . 2))
        (G0017 G0013 G0005))
(G0027 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (BBTOCC . 1) (AATOBB . 2))
        (G0017 G0013 G0005))
(G0028 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (CCTODD . 1) (AATOBB . 2))
        (G0018 G0013 G0005))
(G0029 S0
        ("BBBB" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (CCTODD . 1) (AATOBB . 2))
        (G0018 G0013 G0005))
(G0030 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        SUSPENDED
        ((BBTOCC . 2) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0031 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0032 S0
        ("AAB" "BCCDD")
```

```
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (BBTOCC . 1) (BBTOCC . 1))
        (G0019 G0014 G0005))
(G0033 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        SUSPENDED
        ((BBTOCC . 2) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0034 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0035 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (CCTODD . 1) (BBTOCC . 1))
        (G0020 G0014 G0005))
(G0036 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 2) (BBTOCC . 1) (CCTODD . 1))
        (G0021 G0015 G0005))
(G0037 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 1) (BBTOCC . 1) (CCTODD . 1))
        (G0021 G0015 G0005))
(G0038 S0
        ("AAB" "BCCDD")
        ACTIVE
        T
        ACTIVE
        ((CCTODD . 1) (BBTOCC . 1) (CCTODD . 1))
        (G0021 G0015 G0005))
(G0039 S0
        ("AACC" "CCDD")
        ACTIVE
        T
        ACTIVE
        ((BBTOCC . 2) (CCTODD . 1) (CCTODD . 1))
        (G0022 G0015 G0005))
(G0040 S0
```

```
          ("AAB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((BBTOCC . 1) (CCTODD . 1) (CCTODD . 1))
          (G0022 G0015 G0005))
 (G0041 S0
          ("AAB" "BCCDD")
          ACTIVE
          T
          ACTIVE
          ((CCTODD . 1) (CCTODD . 1) (CCTODD . 1))
          (G0022 G0015 G0005)))
NIL
INTERRUPT MODE!
WHAT NOW:*EV
EV WHAT?
*(EXIT)
(DEFPROP LEFT
 (LAMBDA (X)
  (COND [(NOT (STRINGP X))
         (ERROR "NON-STRING ARGUMENT TO LEFT")]
        [(EMPTYP X) (ERROR "EMPTY STRING TO LEFT")]
        [(SUBSTRING X 1 1)]))
 EXPR)
(DEFPROP REST
 (LAMBDA (X)
  (COND [(NOT (STRINGP X))
         (ERROR "NON-STRING ARGUMENT TO REST")]
        [(EMPTYP X) (ERROR "EMPTY STRING TO REST")]
        [(EQ (STRLEN X) 1) ""]
        [(SUBSTRING X 2 (STRLEN X))]))
 EXPR)
(DEFPROP EMPTYP
 (LAMBDA (X)
  (COND [(NOT (STRINGP X))
         (ERROR "NON-STRING ARGUMENT TO EMPTYP")]
        [(EQSTR X "")]))
 EXPR)
(DEFPROP BEGINS
 (LAMBDA (A B)
  (COND [(NOT (AND [STRINGP A] [STRINGP B]))
         (ERROR "NON-STRING ARGUMENT TO BEGINS")]
        [(EMPTYP A)]
        [(NOT (*GREAT (STRLEN A) (STRLEN B)))
         (EQSTR A (SUBSTRING B 1 (STRLEN A)))]))
 EXPR)
(DEFPROP STRLEN (LAMBDA (X) (LENGTH (EXPLODEC X))) EXPR)
```

J. Plans for the Future

The expansion of this primitive computer system will proceed with a definite plan. In the first stage, comprehensive error-catching facilities will be included. At present, an error during the EVALuation of any LISP expression returns control through

the error package to the top level of LISP. Error checks should keep control in such cases, and should prevent the user from entering nonsense structures to KOTSU.

In the second stage it is planned to include a logic programming capability to function in the test part of a quadruple. The whole system will then become an instrument for investigating possible control structures to accompany these new and powerful devices of logic programming.

In the third and extremely important stage, the system will be made much more responsive to the user's needs. Debugging and tracing aids will be refined, and a smooth and convenient means to communicate with the system will be devised. In the fourth stage existing capabilities of the system will be expanded and refined so that the semantics of KOTSU will be enriched and enlarged, if not deepened.

In the fifth stage we will study appropriate syntactic forms for expressing the kinds of control operations that are a major part of KOTSU. We will be designing specific programming language constructs for controlling and testing the interaction among concurrent world lines.

In the sixth stage we will begin reprogramming of basic parts of the system

taking into account the pointer manipulation facilities available in LISP. Along with these developments we will be organizing a precise definition for the whole system including both the structures and the processes involved.

A major emphasis inthe immediate future will be on developing and recognizing applications of these facilities to interesting computational problems. It has not been possible to include a carefully developed application here, but we can indicate in general and suggestive terms how we can approach a problem in computational linguistics. This brief discussion will focus on the attitude rather than on the technology.

Historically, language has been studied as though it consisted of a system of objects organized in certain ways and processed by a general purpose computer. This description is even true of pre-Chomskyan linguistics. Processes were indeed involved, but were subordinate to the categories and grammatical relations fostered in what is sometimes called "school grammar". The excessiveconcern with automata theory has seems to inflict linguistics and certainly computational linguistics arises from an unquestioning trust in these systems of categories and grammatical relations.

Here we are invited to take a fundamentally different approach to the study of language. And thus also a fundamentally different view of computer implementations of language processes. The facilities described here constitute a systematic and dis-

ciplined framework with which complex interacting processes can be specified and investigated in an experimental setting. It is clearly not a question of being able to specify processes that can't be specified in other means. Rather, we have a completely different way of specifying algorithms, and an invitation to consider algorithms from a novel point of view.

For example, we can consider that the recognition of utterances is not so much a matter of assigning parts of a sequence to grammatical categories and relations, but rather is a matter of bringing a set of concurrent processes into some kind of stable cooperating configuration. Such notions as resonance and entrainment seem apt metaphors. We may understand that the definiteness of a linguistic structure is determined not by the categories and relations that its constituents enter into, but by the nature and degree of cooperation among those constituents. If we cultivate a healthy disrespect for all those background concepts and abstractions we may discover in the foreground that it is the processes of language that should be described by linguistic theories, and it those same processes that may be duplicated with a computer in a way that will preserve the power of natural language, and not simply confront us with a barrage of sterile combinatorial nattering.

For a specific case, we may suggest that language recognition consists in the parallel operation of a whole system of individual recognizers. One identifies mood, another identifies cases, such as actor, agent, instrument, etc. Another recognizer identifies degree of definiteness and quantification. Another searches for "root" or "radical" elements in the utterance. Another determines temporal or "tense" factors. A spectrum of other processes may be involved with such emotional issues as fearing, loathing, desiring, and with propositional attitudes such as hoping, believing, wishing, and so on. In a basic sense the form of cooperation among a family of such preferential judgments is as specified in this report. The substance of it is further specified by the LISP processes and the logical inferences that occur as acts and tests in the quadruples. We will be investigating these matters thoroughly in the immediate future.

Appendix

Here we discuss the role of the technical notion of an occurrence in the computer processing of language text. We indicate that the formal aspect of language processing is devoid of meaning. We try to sketch the beginning of a formal theory of occurrences, show how these ideas may be incorporated into a running computer system, and illustrate the first stages of such a system in application to some English text for defining functions in LISP and to English text taken from the manual for courts martial. A few of the existing functions used in the computer system are described in order to show what kind of programming is involved.

A. Introduction

Natural languages and the languages of computation and logic have two important characteristics in common. They each deal with expressions of some sort, and each provides a systematic way of interpreting its expressions. At the present time, we have no clear indication of the nature of interpretation in natural language. Interpretation in logic is reasonably well-defined, although we never have the last word on any subject. For the case of computation there are interpretive issues that are clear and some that are not so clear. We may interpret a computational transaction as consisting of the manipulation of arrangements of natural numbers, in which case the numbers are introduced purely as a matter of interpretation. Or we may understand that a computation is a sequence of functional applications, this sequence required to meet certain formal conditions. In this case, functions and functional application constitute the most basic interpretation. We may interpret a computation as a sequence of physical events of a specified kind, again required to meet certain formal conditions. There are other interpretations of computation, but each of them makes use of an operation of substitution. Expressions or things get replaced by other expressions or things in a systematic way. Substitution seems to appear in all formal treatments of natural language, computation and logic. But aside from substitution there appears to be no other unifying conception in the theory of computation.

There is, however, another level of order that is prior to the level in which a substitution operation is specifiable. It is prior in the sense that substitution cannot be well-defined without reference to the objects and relations that appear in this more elemental level of structure. At this level we speak of occurrences and of complex arrangements of occurrences. We specify what can be done to occurrences, and what can be done with them. Once the regularity in this level of structure has been described, we can then introduce the substitution operation, in which an occurrence of one thing is substituted for an occurrence of a second in an occurrence of a third. There seems to be no way to formulate the notion of substitution without reference to occurrences and their properties. We may, as has been done in

the past, choose to ignore the structure of pure occurrences on the ground that the notion is far too primitive to yield any concrete benefit. But it is difficult to see how we can know that there will be no benefit unless we look for one.

The studies reported here have concentrated on occurrences, arrangements of occurrences and their manipulation by computer. The computer system which is described and illustrated in later sections deals with occurrences of natural language objects. One very important question in the study of natural language concerns the distinction between the form or structure of linguistic objects and the meaning of these objects. The question is decisively obscured by the fact that we have no clear idea of what these meanings are, how they are arranged, or how they are systematically related to expressions, situations and people. In addition, the conventional view of linguistic form is not completely neutral with regard to the question of meaning. For example, well-known grammatical categories such as noun phrases or relative clauses provide a complex system of structures for later association with meanings. The fact is, however, that the concept of a noun phrase is a concept in part about meanings, and not about form only. Noun phrases have a nominal function, hence a referential function and hence an interpretive character. Since these categories and others like them are not independent of meaning, they are unsuited for the representation of pure uninterpreted form in any language. Moreover, the determination of specific categories is understood, at least in linguistic studies, to be a matter of empirical observation, so that even the concept of categories is dependent upon an expected interpretation. Linguistic data fully justify this expectation, I think, but the description of the uninterpreted form of language may still not make use of these categories.

An adequate representation of form is provided by the study of occurrences. In these pages I will examine this idea informally, and indicate, toward the end, what one might do to demonstrate adequacy in a convincing way.

In the next section we very briefly review some of the ideas that have appeared in computational linguistics, and we observe that the present study does not make important use of those ideas. Existing studies in computational linguistics may be interesting in themselves, and may on occasion have practical application. But there is no objective evidence that these studies have anything to do with natural language, except that similar (but not identical) alphabets of symbols are involved at selected levels of organization. The studies reported here are moving in a different direction. I believe that it may be valuable to discover how naturally occurring living things implement their languages. Concrete benefits may follow from such discovery, and I will later suggest at least one such benefit, however modest.

In subsequent sections there will be a first approximation to a theory of occurrences, an indication of the difference between occurrences and occurrences <u>of</u> things, and an attempt to

show how the characteristics of occurrence structure are manifested in natural language. Then we describe a computer system for dealing with occurrences, illustrate its application to natural language, and suggest possible implications for the development of a truly universal programming language.

B. Computer Linguistics

Computer processing of natural language text originally focussed on the representation of the text provided in terms of such conventional categories as nouns, verbs, noun phrases, prepositional phrases, relative clauses, and others. Exactness was achieved by formalizing this treatment in terms of phrase-structure grammars as originally defined by Chomsky. The whole class of phrase-structure grammars was separated by Chomsky into four subclasses, each of which was shown to be related to a certain kind of mechanical recognition device. For a period of time it was believed that adequate implementation of these recognition devices might provide computer systems with the ability to exhibit interesting and useful linguistic capacities. The judgment that the recognition devices which Chomsky associated with his grammars in some important way reflected the realities of human language. was never, however, supported by objective evidence or argument. There is hardware associated with human language - the brain, the oral cavity and supporting muscles, and the pulmonary system - but one cannot really say that language makes use of a pushdown store until one has found that pushdown store in the hardware.

A second major effort in natural language by computer was initiated by a series of studies many of which were done at M.I.T. during the nineteen sixties. This work flowered in Winograd's famous study which exploited computer handling of natural language in a well-defined and manageable setting. Among other things, this study may be interpreted as providing evidence that the human language capacity is organized so much like a program-controlled computation that the subroutine concept itself is a central organizing principle in human language. Moreover, Winograd's system was organized in such a way that the process of constructing underlying meanings was intimately bound up with the more superficial stages of syntactic parsing. Such substructures as words, phrases and clauses were indeed constructed and assigned meanings, but the involvement of semantic analysis in the whole process was thought to show that an independent parsing mechanism could not be factored out of language. But since language processes are factored according to the subroutine concept, and since language and thought are inseparable, thought processes must also be factored according to the subroutine concept. In short, the human mind must be a computer.

This is a fine argument, and I think it is largely correct, but I would suggest one emendation. Meanings can be eliminated altogether, and the argument still stands. That is, the <u>form</u> of mental activity is computational in character. But it is inappropriate to suggest that all aspects of mental activity are

computational. Science has nothing whatever to offer with respect to mind or to meanings, and for that reason alone it is inappropriate, in a scientific setting, to determine that meanings have a certain form, whether that is the form of a program, a data base, a deduction or any other kind of computational structure. There is, at the moment, no way to base conclusions within a theory of mind on experimental observation, because we simply do not know what to observe. And it is confusing to suggest that the meaning of an expression is determined by how it is "used". In the first place this proposal suggests that material purposeful "use" exhausts or essentially determines the significance of linguistic expressions. And in the second place it makes the unjustified assumption that the phrase "how it is used" is itself precise and definite in significance, whereas this phrase may be little more than a slogan, whose principal effect is to inhibit the asking and answering of interesting and important questions about human language.

If we regard computational linguistics as a purely technological and artifactual activity, then of course it makes no difference what relation it has to the study of human language. On the other hand, biological systems have shown considerable success at doing what they do, and at changing their form to meet changing requirements. It may be appropriate to study human language in a non-technological framework, one which makes no unwarranted presuppositions about "practical" utility, or about how "meanings" are represented. It may be appropriate to study human language with reference to structures that are completely neutral with respect to interpretation.

The present studies and the computer implementation that has grown out of them do not fit with the tradition or with the current paradigm in computational linguistics, and it may therefore be helpful to list the attitudes that underly this work.

We reject the idea that syntactic and semantic analysis must proceed in tandem. There is an extensive body of literature in the brain sciences which shows that in fact these functions are indeed factored by the hardware. Brain disorders of one kind leave the grammatical function fairly intact, while those of another kind severely impair grammatical function without seriously affecting any observable intellectual process. (There is some reason to believe, on the basis of these studies, that language is not nearly so implicated with intelligence as has been supposed in the past, although it is surely implicated with mind.)

We reject the idea that a context-free phrase-structure grammar is the appropriate formal vehicle for the description of linguistic structure. A far simpler mechanism may be involved, one which simply inserts the parentheses into a sequence of atomic objects. That is, this mechanism determines which sequences of linguistic units are subsequences of which others, but does not characterize them in terms of grammatical categories. As a consequence, rewrite rules disappear, along with pushdown stores and all the other tedious paraphernalia of automata theory.

But we note that parentheses may still have interpretive significance. The concept of functional application is indicated in LISP by a pair of parentheses. Computation can be defined without the notion of functional application, so that this notion is interpretive, and so therefore is the use of parentheses to indicate it. In LISP functional application is what is being talked about. If there is such a thing, "pure" LISP is meaningful. Bound variables are also meaningful because they contrast with free variables, and free variables are meaningful because they are proper names. The concepts "free" and "bound" are thus interpretive. Being bound or free is an attribute of what occurs and not of occurrence itself. We may then ask to what extent can computations be organized without reference to variables or to objects which are like variables. To the extent that computation can be so organized we will have a computational mechanism without variables, and hence without identifiers. Turing machines have this property, but it may be of interest to see exactly what can be accomplished without sacrificing it.

We reject the idea that the structure of language depends upon what is being talked about. In order to see how far we can go, we omit any treatment of knowledge representation. This study is thus exclusively concerned with the processing by computer of structures that are intended to be like the structures that arise in normal human language, structures characterized in terms of occurrences only.

We adopt the idea that it is important to identify the most general and most fundamental level of structure at which language operates. If there really is a mind, and if that mind is essentially involved with language, and if the brain is implicated as well, then we must look for and find structures in mind, language and brain which can be related in a natural and systematic way. We do not know very much about these structures and these relationships, but we do know this: there is one kind of structure common to all three - the structure of occurrences.

We adopt the idea that significant language processing can be achieved without specifying any particular topic. The evidence is quite clear that there are many different ways of defining linguistic acceptability, and we can even find a level at which nonsense text and uninterpretable text are accepted, where the accepted text still meets stringent regularity conditions. As an example of the former we have "With mergies we sponder a niddling long before the gleeber pluded.", and for the latter we have "Of the apple by which a dog who lost her essay.".

We adopt the idea that it is important to consider nontechnical approaches to those problems facing the Department of Defense. This department is, after all, charged with the awesome responsibility to preserve the physical integrity of the United States, its institutions and its people. In a mere three billion years, natural biological mechanisms have managed to produce the living systems we know, starting with nothing but inorganic structures, and relying on a surprising number of natural defense

mechanisms. In fact, defense and adaptation are obviously quite closely related. Perhaps we can find some reasonably powerful structures and organizing principles among successful biological mechanisms that will contribute positively to the conduct of modern institutional defense. In this study we try to take the first steps toward implementing a computer system for language that is based on what occurs naturally, in the hope that it can evolve into something responsive to human beings. Staff officers are human beings, and additionally are often involved with complex command and control systems with a strong technological orientation. It may be helpful for them to have a language system that is as sensitive to them as it is to the technology. If these technological systems cannot accommodate the living things they affect at least to the same extent that these living systems accommodate each other, then it may be valuable to document this inadequacy and provide compensating adjustments where possible.

Finally, we find that a clear separation of form from interpretation helps to distinguish the purely formal aspect of language from the purely interpretive aspect. It also provides a well-defined framework within which we can ask if some aspect of language depends essentially upon both. For example, we have seen that functional application is found in language as a result of providing a systematic interpretation. The attribution of a property to an object is reflected in the form of language as well as in its interpretation. In the phrase "The girl who has blue eyes" a complete predication is contained in a modifying feature. The modifying feature is "who has blue eyes" and the predication is "girl has blue eyes". Cases like this one (and they are plentiful) are interesting because the phrase "who has blue eyes" is thought to be structurally subordinate to "the girl". We do not yet understand how subordination, which is purely formal, and attribution, which is interpretive, are systematically related. The interplay of form and interpretation is ill understood in this case as well as in many others like it. And this issue is surely of the first importance in studying the possibility of using a computer for natural language.

C. Theory of Occurrences

When I speak of occurrences I have in mind the kind of thing referred to when we write that an occurrence of a variable "y" in "A" is free or bound in "(FA)" according as it is free or bound in "A". The definitions of free and bound refer to occurrences of variables and not simply to variables. In a Turing machine formulation of computability theory, there are occurrences of marks on tape squares, and indeed, there are occurrences of the events of marking, erasing, and shifting left or right. The fact that we can speak of occurrences of objects and of events suggests that there may be something interesting to say about occurrences in abstraction from what they are occurrences of. It will be important to bear in mind, throughout this report, that we speak of occurrences only, and we mean specifically to exclude any characteristics they might have that derive from what they are occurrences of.

The theoretical machinery will deal with individual occurrences and with finite sets of occurrences, where these finite sets will have defined on their members relations which are characteristic of occurrences. The fundamental relation on a finite set of occurrences we call an immediate dominance relation, ID, if it is irreflexive, not necessarily symmetric and not necessarily transitive. No occurrence can ever immediately dominate itself, but pairs of occurrences may but need not immediately dominate each other. And it may or may not be the case that x ID z when x ID y and y ID z. Intuitively, the ID relation ought to formalize the immediate constituency relation for expressions in a language. It ought to formalize the relation between a composite process and its immediate subprocesses and it ought to formalize the relation holding between things that are serially ordered. The notion of immediate dominance is decisively directional. If x ID y then we say that x immediately dominates y, and that y is immediately dominated by, or is subordinate to, x. In general, ID is simply an irreflexive relation.

There is also considerable importance to a certain transitive relation derived from ID. The transitive closure, trans(R), of R, is defined so that it includes R, contains the pair (x, z) whenever it contains both (x, y) and (y, z), and contains nothing else. If S is a (finite) set of occurrences with an ID relation, and R is trans(ID), if there is some x' in S such that for all y in S different from x', x' R y, then x' is a principal occurrence of the set S, and an arrangement is a triple consisting of the set S, the relation R, and a specified principal occurrence in S. The relation R is called the dominance relation, and if x R y we say that x dominates y and that y is dominated by or is subordinate to x. In an arrangement, occurrences may or may not dominate themselves, including the principal occurrence.

An aggregate of occurrences is a superficial kind of structure. But the dominance relation has an integrating effect, since it permits the distinction between those sets of occurrences that have principal occurrences from those that do not. It allows us to distinguish arrangements from mere piles. We note further that the dominance relation is introduced at the cost of using the notion of a transitive closure, a notion that is recursive in character. In short, the role played by recursive definitions in the notion of well-formation is enacted at the level of occurrences, before any specification is made about what the expressions mean, and indeed even before any specification is made about what is to occur. Arrangements of events also have this character.

The dominance relation is not irreflexive, since if x ID y and y ID x, then x R y and y R x and then x R x. But the dominance relation need not be symmetric and certainly need not be weakly anti-symmetric. It is simply a certain kind of transitive relation. And a set of occurrences whose dominance relation determines a principal occurrence is an arrangement.

An arrangement (S, R, p) may have a subarrangement (S*, R*, a*) where S* is included in S, R* is included in R and (S*, R*, a*) is itself an arrangement. By convention, certain subarrangements of an arrangement, or of a class of arrangements, will be called features. A feature of an arrangement S is a subarrangement S* of S none of whose members dominate any occurrences outside S* and such that if any occurrence outside S* dominates any occurrence in S*, it also dominates the principal occurrence in S*. In short, a feature is a subarrangement whose only connection with the rest of the arrangement is mediated through its principal occurrence. The dominance relations outside and inside S* are connected solely at the principal occurrence of S*.

In discussing and exemplifying arrangements we will often use simple identifiers in place of features. These identifiers are not really part of the structure of occurrences, but are used only to make their presentation more compact and perhaps more palatable. They also encourage recognition of the same arrangement of occurrences in different settings. In this connection we note that by the most basic property of occurrences, no two arrangements can be completely identical because the occurrences they are made of are all distinct. But their relations may be "isomorphic" in a well-defined sense which we will not pause to elaborate here. (And obviously, we can also define the notion that there is a homomorphism from one arrangement to another, in an obvious way.) The identity relation among arrangements will play an important part in future developments in these studies.

We may hazard an informal comment that the study of the non-trivial aspects of occurrences and of arrangements is the study of pure uninterpreted form. We expect to show that computability theory will be expressible with reference only to the manipulation of arrangements. We may also speculate that the significance of arrangements of occurrences in the brain may be determined by the different modalities which characterize the things that occur in an arrangement. Patterns of modalities of regular and re-occurring forms may also be identifiable in a framework making reference to arrangements and further elaborations of these ideas.

If we try to devise expressions to stand only for occurrence structure then no two occurrences bear the same name or the same "position" in the expression, reflecting the fact that no two occurrences are the same. Occurrences are decisively and uniquely distinguished from their environment. Whatever the nature of the space they are in, occurrences are decisively differentiated out of that space, and it is this sharp differentiation that we refer to when we say that an object is discrete. Of course, a set of occurrences may have an ID relation on it but that relation may happen to be empty. We may represent the transitive closure of ~the immediate dominance relation in some suitable notation. Then a may immediately dominate f, f may immediately dominate b and g and g may immediately dominate e. Then the transitive closure of this relation will be

a(f, b, g, e)
f(g, e)
g(e)

where x(yl, ..., yM) means that x dominates yi, for i in the appropriate range.

By inspection we can see that a set of occurrences may) lack a principal occurrence and so is not an arrangement. A subarrangement describe the top-down structure of a whole arrangement, where the term "top-down" is to be taken with reference to the (directed) dominance relation. Subarrangements may be understood as features, and we see that features may contain other features as parts. We agree to restrict this "part" relation to subarrangements, and note that an occurrence standing alone is an arrangement.

Occurrences of things have different properties than do occurrences. For example, if we want to combine two arrangements to make a new single arrangement, we may take a set union and then add what is needed to the dominance relation. We can substitute one feature for another in an arrangement simply by deleting one subarrangement and adding another.

In many formal languages, expressions are built on an occurrence structure in which all subarrangements are features, so that a substitution operation is readily specifiable in simple terms. In other formalisms substitution is not so simple a matter. In the lambda calculus, for example, substitution of occurrences is allowed or prohibited only with reference to conditions that are determined by what occurs. If we chose to set conditions on substitution in arrangements, we might ignore the properties of the things that occur. We might require, for example, that if every subarrangement is a feature before the substitution, then the same must hold true after the substitution. More generally we might require that arbitrarily specified properties of a set of arrangements ordered by the part relation must be invariant under transformations. Such is the nature of the restrictions on substitution in the lambda calculus. It may be the case that interesting variations on substitution may turn up as we elaborate further the theory of occurrences.

D. Examples of Computer Processing

The iexamples we can present at this time demonstrate that it is possible to accomplish a considerable amount of purely formal or syntactic analysis provided we forego the use of grammatical categories, noting that they are hardly necessary in any event. We will begin by displaying the analysis, as it is represented in the system, of one sequence of English text that describes a string manipulation function to a recursive language like LISP.

>Let x be the only argument to the function f. The value of f, when x is zero, is one. Otherwise : the value of f is the product of x with (f(subl x)).

```
((G0323
  CONSEC
  ((G0325 FEATURES (#DOT . T))
   (G0327
    SERIAL
    ((G0241
      CONSEC
      ((G0221 FEATURES (Let . T))
       (G0223 FEATURES (x . T))
       (G0225 FEATURES (be . T))
       (G0227 FEATURES (the . T))
       (G0229 FEATURES (only . T))
       (G0231 FEATURES (#FRAME . T) (argument . T))
       (G0329
        CONSEC
        ((G0235 FEATURES (the . T))
         (G0237 FEATURES (function . T))
         (G0239 FEATURES (f . T)))
        FEATURES
        (#PREP . T)
        (to . T))))))))
 (G0317
  CONSEC
  ((G0319 FEATURES (#DOT . T))
   (G0321
    SERIAL
    ((G0269
      CONSEC
      ((G0243 FEATURES (The . T))
       (G0245 FEATURES (#FRAME . T) (value . T))
       (G0331 CONSEC
              ((G0249 FEATURES (f . T)))
              FEATURES
              (#PREP . T)
              (of . T))))
     (G0271 FEATURES (#COM . T))
     (G0265
      CONSEC
      ((G0251 FEATURES (when . T))
       (G0253 FEATURES (x . T))
       (G0255 FEATURES (is . T))
       (G0257 FEATURES (zero . T))))
     (G0267 FEATURES (#COM . T))
     (G0263
      CONSEC
      ((G0259 FEATURES (is . T)) (G0261 FEATURES (one .
T))))))))
 (G0311
  CONSEC
  ((G0313 FEATURES (#DOT . T))
   (G0315
    SERIAL
    ((G0307 CONSEC ((G0273 FEATURES (Otherwise . T))))
     (G0309 FEATURES (#COLN . T))
     (G0305
```

```
CONSEC
((G0275 FEATURES (the . T))
 (G0277 FEATURES (#FRAME . T) (value . T))
 (G0333
  CONSEC
  ((G0281 FEATURES (f . T))
   (G0283 FEATURES (is . T))
   (G0285 FEATURES (the . T))
   (G0287 FEATURES (product . T))
   (G0335
    CONSEC
    ((G0291 FEATURES (x . T))
     (G0337
      CONSEC
      ((G0303
        CONSEC
        ((G0295 FEATURES (f . T))
         (G0301
          CONSEC
          ((G0297 FEATURES (subl . T))
           (G0299 FEATURES (x . T)))))))
       FEATURES
       (#PREP . T)
       (with . T)))
     FEATURES
     (#PREP . T)
     (of . T)))
   FEATURES
   (#PREP . T)
   (of . T)))))))))
```

This display shows the result of "parsing" the original sequence, including the treatment of prepositions. Symbols such as the colon, semicolon, comma, and certain others, are parsed as the outermost, or global, symbols indicating grouping. All groupings run from the occurrence of the symbol to the next grouping indicator to the right, and the hierarchical subordination is established as the input sequence is scanned from left to right. After this parse is completed, the system parses again for prepositions,

I will use the word "incorporation" to refer to what is constructed in the computer system in place of the word "representation" since the latter word may suggest that the analysis presupposes what is represented. A complete list of what is incorporated in the present system would include: occurrence identifiers, grouping indicated by periods, commas and other punctuation marks, grouping indicated by pauses, hesitations, etc., grouping indicated by prepositions, different kinds of ordering indicated by SERIAL, CONSEC and UNORDERED, the feature convention, and finally the identification of frame elements. All the linguistic material that is subject to direct interpretation is incorporated as features. Each feature is incorporated as a dotted pair whose cdr is T. In the future, the cdr of a feature may be an expression suitable for evaluation or a func-

tion suitable for application. These items will affect the significance of the arrangements they are a part of. The special pause characters that indicate bracketing are of some importance. These are inserted by the user into the input sequence in natural places, much as a patient teacher pauses frequently as he tries to articulate a complex idea. These pauses are treated exactly as punctuation marks by the scanner. The only one shown in the examples displayed here is the colon, ":", which is therefore not being used here as a colon. In command and control applications we visualize separate pause and bracketing keys on the system communication devices. We expect to devote some attention to these pause indicators in future work, and the additional grouping developed thus is indicated. The next display excludes the generated occurrence symbols which are explicitly stored in the computer. The next display eliminates the ordering indicators, SERIAL, CONSEC and UNORDERED.

```
((CONSEC
  ((FEATURES (#DOT . T))
   (SERIAL
    ((CONSEC
      ((FEATURES (Let . T))
       (FEATURES (x . T))
       (FEATURES (be . T))
       (FEATURES (the . T))
       (FEATURES (only . T))
       (FEATURES (#FRAME . T) (argument . T))
       (CONSEC
        ((FEATURES (the . T))
         (FEATURES (function . T))
         (FEATURES (f . T)))
        FEATURES
        (#PREP . T)
        (to . T))))))))
 (CONSEC
  ((FEATURES (#DOT . T))
   (SERIAL
    ((CONSEC
      ((FEATURES (The . T))
       (FEATURES (#FRAME . T) (value . T))
       (CONSEC ((FEATURES (f . T))) FEATURES (#PREP . T)
(of . T))))
     (FEATURES (#COM . T))
     (CONSEC
      ((FEATURES (when . T))
       (FEATURES (x . T))
       (FEATURES (is . T))
       (FEATURES (zero . T))))
     (FEATURES (#COM . T))
     (CONSEC ((FEATURES (is . T)) (FEATURES (one .
T))))))))
 (CONSEC
  ((FEATURES (#DOT . T))
   (SERIAL
    (.CONSEC ((FEATURES (Otherwise . T))))
```

```
                (FEATURES (#COLN . T))
                (CONSEC
                 ((FEATURES (the . T))
                  (FEATURES (#FRAME . T) (value . T))
                  (CONSEC
                   ((FEATURES (f . T))
                    (FEATURES (is . T))
                    (FEATURES (the . T))
                    (FEATURES (product . T))
                    (CONSEC
                     ((FEATURES (x . T))
                      (CONSEC
                       ((CONSEC
                         ((FEATURES (f . T))
                          (CONSEC ((FEATURES (subl . T)) (FEATURES
(x . T)))))))
                        FEATURES
                        (#PREP . T)
                        (with . T)))
                       FEATURES
                       (#PREP . T)
                       (of . T)))
                     FEATURES
                     (#PREP . T)
                     (of . T)))))))))
          ((((FEATURES (#DOT . T))
             (((((FEATURES (Let . T))
                 (FEATURES (x . T))
                 (FEATURES (be . T))
                 (FEATURES (the . T))
                 (FEATURES (only . T))
                 (FEATURES (#FRAME . T) (argument . T))
                 (((FEATURES (the . T))
                   (FEATURES (function . T))
                   (FEATURES (f . T)))
                  FEATURES
                  (#PREP . T)
                  (to . T)))))))
           (((FEATURES (#DOT . T))
             (((((FEATURES (The . T))
                 (FEATURES (#FRAME . T) (value . T))
                 (((FEATURES (f . T))) FEATURES (#PREP . T) (of .
T))))
                (FEATURES (#COM . T))
                (((FEATURES (when . T))
                  (FEATURES (x . T))
                  (FEATURES (is . T))
                  (FEATURES (zero . T))))
                (FEATURES (#COM . T))
                (((FEATURES (is . T)) (FEATURES (one . T)))))))
           (((FEATURES (#DOT . T))
             (((((FEATURES (Otherwise . T))))
               (FEATURES (#COLN . T))
               (((FEATURES (the . T))
                 (FEATURES (#FRAME . T) (value . T))
```

```
                (((FEATURES (f . T))
                  (FEATURES (is . T))
                  (FEATURES (the . T))
                  (FEATURES (product . T))
                  (((FEATURES (x . T))
                    (((((FEATURES (f . T))
                        (((FEATURES (subl . T)) (FEATURES (x .
T)))))))
                     FEATURES
                     (#PREP . T)
                     (with . T)))
                   FEATURES
                   (#PREP . T)
                   (of . T)))
                 FEATURES
                 (#PREP . T)
                 (of . T)))))))
```

We notice here that a consecutive sequence of words that constitutes a prepositional phrase in conventional notation has two features to indicate that fact. One feature, #PREP, indicates that it is is prepositional phrase, and the other feature indicates directly which preposition is present. This uniform treatment is adopted because we want to avoid any structure in the incorporation that suggests any particular interpretation. The following incorporation shows only the features themselves. This facility will become more valuable as the system is developed and features become more complex.

```
          (((((#DOT . T))
             ((((((Let . T))
                 ((x . T))
                 ((be . T))
                 ((the . T))
                 ((only . T))
                 ((#FRAME . T) (argument . T))
                 ((((the . T)) ((function . T)) ((f . T)))
                  (#PREP . T)
                  (to . T))))))))
           ((((#DOT . T))
             ((((((The . T)) ((#FRAME . T) (value . T))
                             ((((f . T))) (#PREP . T) (of . T))))
               ((#COM . T))
               ((((when . T)) ((x . T)) ((is . T)) ((zero . T))))
               ((#COM . T))
               ((((is . T)) ((one . T)))))))
           ((((#DOT . T))
             ((((((Otherwise . T))))
               ((#COLN . T))
               ((((the . T))
                 ((#FRAME . T) (value . T))
                 ((((f . T))
                   ((is . T))
                   ((the . T))
                   ((product . T))
```

```
                    ((((x . T))
                      ((((((f . T)) ((((subl . T)) ((x . T)))))))
                        (#PREP . T)
                        (with . T)))
                       (#PREP . T)
                       (of . T)))
                     (#PREP . T)
                     (of . T)))))))))
```

This last display removes the identification #FEATURE and displays the words as they appear in the bracketed form. Transformations would need to be applied to these to bring the prepositions into the proper positions and to insert the indicated punctuation marks. Aside from these two quite standard operations, output text will be generated from the internal incorporations simply by dropping all the parentheses - that is, by ignoring the hierarchical order in the incorporation.

```
          ((((#DOT)
             (((((Let) (x)
                       (be)
                       (the)
                       (only)
                       (#FRAME argument)
                       (((the) (function) (f)) #PREP to)))))))
           (((#DOT)
             (((((The) (#FRAME value) (((f)) #PREP of)))
               (#COM)
               (((when) (x) (is) (zero)))
               (#COM)
               (((is) (one)))))))
           (((#DOT)
             (((((Otherwise)))
               (#COLN)
               (((the)
                 (#FRAME value)
                 (((f) (is)
                       (the)
                       (product)
                       (((x) (((((f) (((subl) (x)))))) #PREP
with)) #PREP of))
                  #PREP
                  of)))))))
```

Here, by way of further example, is the computer analysis of a definition of a function on strings. It includes the operation of a frame feature. As the individual elements in the sequence are scanned by the grouping routines, they are checked against a fixed list of items which are presumed to have significance, and from which the significance of the whole is to be derived. The frame words identified here include: strings, arguments,return, truth-value, truth, value, falsehood, length, else, begins, rest. There are others used elsewhere. These are all items whose significance will have to be "known" to the system before it can

"know" the function described by this text. Their role in the later stages of the system's development will be to excite the application of specific transformations to arrangements where the principal technical issues requiring attention will include scope and constituency.

>Occursin takes two strings, A and B, as arguments : and returns a truth-value as its value. We ask if the length of A is greater than the length of B. If so, the value returned is falsehood. Else, if A begins B : then the value returned is truth. Otherwise we ask recursively if A occursin the rest of B.

```
((G0559
  CONSEC
  ((G0561 FEATURES (#DOT . T))
   (G0563
    SERIAL
    ((G0425
      CONSEC
      ((G0383 FEATURES (Occursin . T))
       (G0385 FEATURES (takes . T))
       (G0387 FEATURES (two . T))
       (G0389 FEATURES (#FRAME . T) (strings . T))))
     (G0427 FEATURES (#COM . T))
     (G0421
      CONSEC
      ((G0391 FEATURES (A . T))
       (G0393 FEATURES (and . T))
       (G0395 FEATURES (B . T))))
     (G0423 FEATURES (#COM . T))
     (G0417
      CONSEC
      ((G0565 CONSEC
              ((G0399 FEATURES (#FRAME . T) (arguments .
T)))
              FEATURES
              (#PREP . T)
              (as . T))))
     (G0419 FEATURES (#COLN . T))
     (G0415
      CONSEC
      ((G0401 FEATURES (and . T))
       (G0403 FEATURES (#FRAME . T) (returns . T))
       (G0405 FEATURES (a . T))
       (G0407 FEATURES (#FRAME . T) (truth-value . T))
       (G0567
        CONSEC
        ((G0411 FEATURES (its . T))
         (G0413 FEATURES (#FRAME . T) (value . T)))
        FEATURES
        (#PREP . T)
        (as . T)))))))
  (G0553
   CONSEC
   ((G0555 FEATURES (#DOT . T))
```

```
(G0557
 SERIAL
 ((G0457
   CONSEC
   ((G0429 FEATURES (We . T))
    (G0431 FEATURES (ask . T))
    (G0433 FEATURES (if . T))
    (G0435 FEATURES (the . T))
    (G0437 FEATURES (#FRAME . T) (length . T))
    (G0569
     CONSEC
     ((G0441 FEATURES (A . T))
      (G0443 FEATURES (is . T))
      (G0445 FEATURES (greater . T))
      (G0447 FEATURES (than . T))
      (G0449 FEATURES (the . T))
      (G0451 FEATURES (#FRAME . T) (length . T))
      (G0571 CONSEC
             ((G0455 FEATURES (B . T)))
             FEATURES
             (#PREP . T)
             (of . T)))
     FEATURES
     (#PREP . T)
     (of . T))))))))
(G0547
 CONSEC
 ((G0549 FEATURES (#DOT . T))
  (G0551
   SERIAL
   ((G0475 CONSEC
           ((G0459 FEATURES (If . T)) (G0461 FEATURES
(so . T))))
    (G0477 FEATURES (#COM . T))
    (G0473
     CONSEC
     ((G0463 FEATURES (the . T))
      (G0465 FEATURES (#FRAME . T) (value . T))
      (G0467 FEATURES (#FRAME . T) (returned . T))
      (G0469 FEATURES (is . T))
      (G0471 FEATURES (#FRAME . T) (falsehood .
T)))))))))
(G0541
 CONSEC
 ((G0543 FEATURES (#DOT . T))
  (G0545
   SERIAL
   ((G0507 CONSEC ((G0479 FEATURES (Else . T))))
    (G0509 FEATURES (#COM . T))
    (G0503
     CONSEC
     ((G0481 FEATURES (if . T))
      (G0483 FEATURES (A . T))
      (G0485 FEATURES (#FRAME . T) (begins . T))
      (G0487 FEATURES (B . T))))
```

```
      (G0505 FEATURES (#COLN . T))
      (G0501
       CONSEC
       ((G0489 FEATURES (then . T))
        (G0491 FEATURES (the . T))
        (G0493 FEATURES (#FRAME . T) (value . T))
        (G0495 FEATURES (#FRAME . T) (returned . T))
        (G0497 FEATURES (is . T))
        (G0499 FEATURES (#FRAME . T) (truth . T))))))))
(G0535
 CONSEC
 ((G0537 FEATURES (#DOT . T))
  (G0539
   SERIAL
   ((G0533
     CONSEC
     ((G0511 FEATURES (Otherwise . T))
      (G0513 FEATURES (we . T))
      (G0515 FEATURES (ask . T))
      (G0517 FEATURES (recursively . T))
      (G0519 FEATURES (if . T))
      (G0521 FEATURES (A . T))
      (G0523 FEATURES (occursin . T))
      (G0525 FEATURES (the . T))
      (G0527 FEATURES (#FRAME . T) (rest . T))
      (G0573 CONSEC
             ((G0531 FEATURES (B . T)))
             FEATURES
             (#PREP . T)
             (of . T)))))))))
```

In the next extension of the system, we will provide simple means to permit the user to introduce those words or items that are assumed to be available in advance. These items define the frame within which the text is to be understood.

We have also processed a number of English sentences taken from the manual of courts martial. This language ought to be as precise as any English text, and its analysis by computer may on occasion be representative of what must be done in other situations. A certain strength in the processes is revealed by these analyses, since they take very little time, and some of the sentences processed were quite large. We give the first in its full display.

>The actions and demeanor of the accused : as observed by the court : or the bare assertion from a reliable source : that the accused is believed to lack mental capacity : or is mentally irresponsible : may be sufficient to warrant inquiry by the court.

```
((G0327
  CONSEC
  ((G0329 FEATURES (#DOT . T))
   (G0331
```

```
                SERIAL
                ((G0323
                  CONSEC
                  ((G0221 FEATURES (The . T))
                   (G0223 FEATURES (actions . T))
                   (G0225 FEATURES (and . T))
                   (G0227 FEATURES (demeanor . T))
                   (G0333
                    CONSEC
                    ((G0231 FEATURES (the . T)) (G0233 FEATURES
(accused . T)))
                    FEATURES
                    (#PREP . T)
                    (of . T))))
                 (G0325 FEATURES (#COLN . T))
                 (G0319
                  CONSEC
                  ((G0335
                    CONSEC
                    ((G0237 FEATURES (observed . T))
                     (G0337
                      CONSEC
                      ((G0241 FEATURES (the . T)) (G0243 FEATURES
(court . T)))
                      FEATURES
                      (#PREP . T)
                      (by . T)))
                     FEATURES
                     (#PREP . T)
                     (as . T))))
                  (G0321 FEATURES (#COLN . T))
                  (G0315
                   CONSEC
                   ((G0245 FEATURES (or . T))
                    (G0247 FEATURES (the . T))
                    (G0249 FEATURES (bare . T))
                    (G0251 FEATURES (assertion . T))
                    (G0339
                     CONSEC
                     ((G0255 FEATURES (a . T))
                      (G0257 FEATURES (reliable . T))
                      (G0259 FEATURES (source . T)))
                     FEATURES
                     (#PREP . T)
                     (from . T))))
                   (G0317 FEATURES (#COLN . T))
                   (G0311
                    CONSEC
                    ((G0261 FEATURES (that . T))
                     (G0263 FEATURES (the . T))
                     (G0265 FEATURES (accused . T))
                     (G0267 FEATURES (is . T))
                     (G0269 FEATURES (believed . T))
                     (G0341
                      CONSEC
```

```
                ((G0273 FEATURES (lack . T))
                 (G0275 FEATURES (mental . T))
                 (G0277 FEATURES (capacity . T)))
                FEATURES
                (#PREP . T)
                (to . T))))
              (G0313 FEATURES (#COLN . T))
              (G0307
               CONSEC
               ((G0279 FEATURES (or . T))
                (G0281 FEATURES (is . T))
                (G0283 FEATURES (mentally . T))
                (G0285 FEATURES (irresponsible . T))))
              (G0309 FEATURES (#COLN . T))
              (G0305
               CONSEC
               ((G0287 FEATURES (may . T))
                (G0289 FEATURES (be . T))
                (G0291 FEATURES (sufficient . T))
                (G0343
                 CONSEC
                 ((G0295 FEATURES (warrant . T))
                  (G0297 FEATURES (inquiry . T))
                  (G0345
                   CONSEC
                   ((G0301 FEATURES (the . T)) (G0303 FEATURES
(court . T)))
                   FEATURES
                   (#PREP . T)
                   (by . T)))
                 FEATURES
                 (#PREP . T)
                 (to . T)))))))))
```

Exactly the same parsing machinery was used for this example as for the others. Here we give the longest sentence tried so far, in two versions - first in its full representation, and then as a system of features (which appearhere as dotted pairs). indicated. This example makes use of the pause indicator ":".

>Although the testimony of an expert on mental disorders : as to his observations and opinion with respect to the mental condition of the accused : may be given greater weight than that of a lay witness, a lay witness who is acquainted with the accused : and who has observed his behavior : may testify as to his observations : and may also give such an opinion : as to the general mental condition of the accused : as may be within the bounds : of the common experience and means of observation of men.

```
((G0435
  CONSEC
  ((G0437 FEATURES (#DOT . T))
   (G0439
    SERIAL
    ((G0431
```

```
CONSEC
((G0221 FEATURES (Although . T))
 (G0223 FEATURES (the . T))
 (G0225 FEATURES (testimony . T))
 (G0441
  CONSEC
  ((G0229 FEATURES (an . T))
   (G0231 FEATURES (expert . T))
   (G0443
    CONSEC
    ((G0235 FEATURES (mental . T))
     (G0237 FEATURES (disorders . T)))
    FEATURES
    (#PREP . T)
    (on . T)))
  FEATURES
  (#PREP . T)
  (of . T))))
(G0433 FEATURES (#COLN . T))
(G0427
 CONSEC
 ((G0445
   CONSEC
   ((G0447
     CONSEC
     ((G0243 FEATURES (his . T))
      (G0245 FEATURES (observations . T))
      (G0247 FEATURES (and . T))
      (G0249 FEATURES (opinion . T))
      (G0449
       CONSEC
       ((G0253 FEATURES (respect . T))
        (G0451
         CONSEC
         ((G0257 FEATURES (the . T))
          (G0259 FEATURES (mental . T))
          (G0261 FEATURES (condition . T))
          (G0453
           CONSEC
           ((G0265 FEATURES (the . T))
            (G0267 FEATURES (accused . T)))
           FEATURES
           (#PREP . T)
           (of . T)))
         FEATURES
         (#PREP . T)
         (to . T)))
        FEATURES
        (#PREP . T)
        (with . T)))
     FEATURES
     (#PREP . T)
     (to . T)))
   FEATURES
   (#PREP . T)
```

```
   (as . T))))
(G0429 FEATURES (#COLN . T))
(G0423
 CONSEC
 ((G0269 FEATURES (may . T))
  (G0271 FEATURES (be . T))
  (G0273 FEATURES (given . T))
  (G0275 FEATURES (greater . T))
  (G0277 FEATURES (weight . T))
  (G0279 FEATURES (than . T))
  (G0281 FEATURES (that . T))
  (G0455
   CONSEC
   ((G0285 FEATURES (a . T))
    (G0287 FEATURES (lay . T))
    (G0289 FEATURES (witness . T)))
   FEATURES
   (#PREP . T)
   (of . T))))
(G0425 FEATURES (#COM . T))
(G0419
 CONSEC
 ((G0291 FEATURES (a . T))
  (G0293 FEATURES (lay . T))
  (G0295 FEATURES (witness . T))
  (G0297 FEATURES (who . T))
  (G0299 FEATURES (is . T))
  (G0301 FEATURES (acquainted . T))
  (G0457
   CONSEC
   ((G0305 FEATURES (the . T)) (G0307 FEATURES
(accused . T)))
   FEATURES
   (#PREP . T)
   (with . T))))
(G0421 FEATURES (#COLN . T))
(G0415
 CONSEC
 ((G0309 FEATURES (and . T))
  (G0311 FEATURES (who . T))
  (G0313 FEATURES (has . T))
  (G0315 FEATURES (observed . T))
  (G0317 FEATURES (his . T))
  (G0319 FEATURES (behavior . T))))
(G0417 FEATURES (#COLN . T))
(G0411
 CONSEC
 ((G0321 FEATURES (may . T))
  (G0323 FEATURES (testify . T))
  (G0459
   CONSEC
   ((G0461
     CONSEC
     ((G0329 FEATURES (his . T))
      (G0331 FEATURES (observations . T)))
```

```
                    FEATURES
                    (#PREP . T)
                    (to . T)))
                  FEATURES
                  (#PREP . T)
                  (as . T))))
                (G0413 FEATURES (#COLN . T))
                (G0407
                 CONSEC
                 ((G0333 FEATURES (and . T))
                  (G0335 FEATURES (may . T))
                  (G0337 FEATURES (also . T))
                  (G0339 FEATURES (give . T))
                  (G0341 FEATURES (such . T))
                  (G0343 FEATURES (an . T))
                  (G0345 FEATURES (opinion . T))))
                (G0409 FEATURES (#COLN . T))
                (G0403
                 CONSEC
                 ((G0463
                   CONSEC
                   ((G0465
                     CONSEC
                     ((G0351 FEATURES (the . T))
                      (G0353 FEATURES (general . T))
                      (G0355 FEATURES (mental . T))
                      (G0357 FEATURES (condition . T))
                      (G0467
                       CONSEC
                       ((G0361 FEATURES (the . T))
                        (G0363 FEATURES (accused . T)))
                       FEATURES
                       (#PREP . T)
                       (of . T)))
                     FEATURES
                     (#PREP . T)
                     (to . T)))
                   FEATURES
                   (#PREP . T)
                   (as . T))))
                (G0405 FEATURES (#COLN . T))
                (G0399
                 CONSEC
                 ((G0469
                   CONSEC
                   ((G0367 FEATURES (may . T))
                    (G0369 FEATURES (be . T))
                    (G0471
                     CONSEC
                     ((G0373 FEATURES (the . T)) (G0375 FEATURES
(bounds . T)))
                     FEATURES
                     (#PREP . T)
                     (within . T)))
                   FEATURES
```

```
                    (#PREP . T)
                    (as . T))))
                  (G0401 FEATURES (#COLN . T))
                  (G0397
                   CONSEC
                   ((G0473
                     CONSEC
                     ((G0379 FEATURES (the . T))
                      (G0381 FEATURES (common . T))
                      (G0383 FEATURES (experience . T))
                      (G0385 FEATURES (and . T))
                      (G0387 FEATURES (means . T))
                      (G0475
                       CONSEC
                       ((G0391 FEATURES (observation . T))
                        (G0477 CONSEC
                               ((G0395 FEATURES (men . T)))
                               FEATURES
                               (#PREP . T)
                               (of . T)))
                       FEATURES
                       (#PREP . T)
                       (of . T)))
                      FEATURES
                      (#PREP . T)
                      (of . T)))))))))


          (((((#DOT . T))
             ((((((Although . T))
                 ((the . T))
                 ((testimony . T))
                 ((((an . T))
                   ((expert . T))
                   ((((mental . T)) ((disorders . T))) (#PREP . T)
(on . T)))
                   (#PREP . T)
                   (of . T))))
                ((#COLN . T))
                (((((((his . T))
                      ((observations . T))
                      ((and . T))
                      ((opinion . T))
                      ((((respect . T))
                        ((((the . T))
                          ((mental . T))
                          ((condition . T))
                          ((((the . T)) ((accused . T))) (#PREP .
T) (of . T)))
                         (#PREP . T)
                         (to . T)))
                       (#PREP . T)
                       (with . T)))
                     (#PREP . T)
                     (to . T)))
```

```
                    (#PREP . T)
                    (as . T))))
                  ((#COLN . T))
                  ((((may . T))
                    ((be . T))
                    ((given . T))
                    ((greater . T))
                    ((weight . T))
                    ((than . T))
                    ((that . T))
                    ((((a . T)) ((lay . T)) ((witness . T)))
                     (#PREP . T)
                     (of . T))))
                  ((#COM . T))
                  ((((a . T))
                    ((lay . T))
                    ((witness . T))
                    ((who . T))
                    ((is . T))
                    ((acquainted . T))
                    ((((the . T)) ((accused . T))) (#PREP . T) (with
. T))))
                  ((#COLN . T))
                  ((((and . T)) ((who . T))
                                ((has . T))
                                ((observed . T))
                                ((his . T))
                                ((behavior . T))))
                  ((#COLN . T))
                  ((((may . T))
                    ((testify . T))
                    ((((((his . T)) ((observations . T))) (#PREP . T)
(to . T)))
                     (#PREP . T)
                     (as . T))))
                  ((#COLN . T))
                  ((((and . T)) ((may . T))
                                ((also . T))
                                ((give . T))
                                ((such . T))
                                ((an . T))
                                ((opinion . T))))
                  ((#COLN . T))
                  (((((((the . T))
                        ((general . T))
                        ((mental . T))
                        ((condition . T))
                        ((((the . T)) ((accused . T))) (#PREP . T)
(of . T)))
                       (#PREP . T)
                       (to . T)))
                     (#PREP . T)
                     (as . T))))
                  ((#COLN . T))
                  ((((((may . T))
```

```
                    ((be . T))
                    ((((the . T)) ((bounds . T))) (#PREP . T)
(within . T)))
                   (#PREP . T)
                   (as . T))))
                ((#COLN . T))
                (((((the . T))
                    ((common . T))
                    ((experience . T))
                    ((and . T))
                    ((means . T))
                    ((((observation . T)) ((((men . T))) (#PREP .
T) (of . T)))
                      (#PREP . T)
                      (of . T)))
                   (#PREP . T)
                   (of . T)))))))))
```

E. The programs

The programs were written as LISP functions, and although there are complications, the logic is fairly straightforward. The complications arise because the control characters for the LISP system must be undone and then done up again before and after English text is entered into the system. The functions I-NITABLE and UNINITABLE do this modification.

```
(DEFPROP INITABLE
 (LAMBDA NIL
  (MODCHR 95. (MODCHR 44. NIL))
  (MODCHR 33. (MODCHR 44. NIL))
  (MODCHR 37. (MODCHR 44. NIL))
  (MODCHR 63. (MODCHR 44. NIL))
  (MODCHR 58. (MODCHR 44. NIL))
  (MODCHR 59. (MODCHR 44. NIL))
  (MODCHR 94. (MODCHR 44. NIL)))
EXPR)
(DEFPROP UNINITABLE
 (LAMBDA NIL
  (MODCHR 95. (MODCHR 65. NIL))
  (MODCHR 33. (MODCHR 65. NIL))
  (MODCHR 37. (MODCHR 65. NIL))
  (MODCHR 63. (MODCHR 65. NIL))
  (MODCHR 58. (MODCHR 65. NIL))
  (MODCHR 59. (MODCHR 65. NIL))
  (MODCHR 94. (MODCHR 65. NIL)))
EXPR)
```

The present top level function is BEGIN and it continues to read and process English text until given a left arrow symbol as terminator.

```
(DEFPROP BEGIN
 (LAMBDA NIL
  (SPRINT (PRT-OCC (PREP-PARSE (GORP (READIT)))) 1.))
EXPR)
```

The parse control is set up by the function READIT which calls INITABLE to modify the control characters, prints the prompt character, looks for the terminating symbol, and then sends the sequence of items to PARSE0.

```
(DEFPROP READIT
 (LAMBDA NIL
  (PROG   (DOT AT SENT LEX)
          (SETQ DOT (MODCHR 46. (MODCHR 44. NIL)))
          (SETQ AT (MODCHR 64. (MODCHR 44. NIL)))
          (INITABLE)
          (PROMPT 62.)
          (TERPRI)
          (CLRBFI)
          (SETQ SENT NIL)
   LOOP   (SETQ LEX (READ))
          (COND ((EQ LEX (QUOTE #TERM))
                 (RETURN
                  (PROG1 (PARSE0 SENT)
                         (PROMPT 42.)
                         (MODCHR 46. DOT)
                         (MODCHR 64. AT)
                         (UNINITABLE))))
                (T (SETQ SENT (KONS SENT LEX)) (GO LOOP)))))
EXPR)
```

There are three basic structures representable in the system's forms. These are SERIALly ordered sets of occurrences, UNORDERED sets of occurrences, and CONSECutively ordered sets of occurrences. The difference between SERIAL and CONSECutive order is that the items in a serially ordered set may have other items occurring among them, so long as their relative order is correct. CONSECutive sets must be just that. These three discriminations are easily definable with reference only to occurrence structure, and when we expand the system to deal with more complex levels of language processing we expect to make significant use of these different forms. At present they are unused, but that is only because the system is largely front-end at present. PARSE0 and PARSE1 generate occurrences to reflect the individual items in the input sequence, and the assemblies of these into arrangements which are appropriately ordered. PARSE0 looks for the terminating character, generates occurrence identifiers, and passes the input to PARSE1 which includes the error checks, looks for certain high level bracketing characters which we have not displayed in the examples shown here, looks for the punctuation marks, and handles the processing of subsequences.

```
(DEFPROP PARSE0
 (LAMBDA(S)
  (PROG   (Y Z)
          (COND ((NULL S) (RETURN NIL)))
          (SETQ Y (PARSE1 S))
          (COND
           ((NULL (CDR Y))
            (RETURN
             (#MK-DOMINANT
```

```
                (#MK-OCC)
                (COND ((ATOM (CAR Y)) Y) (T (CAR Y)))
                (QUOTE SERIAL))))
           ((MEMQ (CADR Y) (QUOTE (#EXCL #QUES #DOT)))
            (SETQ Z (PARSE0 (CDDR Y)))
            (COND
             ((NULL Z)
              (RETURN
               (#MK-DOMINANT
                (#MK-OCC)
                (LIST
                 (#ADD-FEATURE (#MK-OCC) T (CADR Y))
                 (#MK-DOMINANT
                  (#MK-OCC)
                  (COND ((ATOM (CAR Y)) (LIST (CAR Y)))
                        (T (CAR Y)))
                  (QUOTE SERIAL)))
                (QUOTE CONSEC))))
             (T
              (RETURN
               (LIST
                (#MK-DOMINANT
                 (#MK-OCC)
                 (LIST
                  (#ADD-FEATURE (#MK-OCC) T (CADR Y))
                  (#MK-DOMINANT
                   (#MK-OCC)
                   (COND ((ATOM (CAR Y)) (LIST (CAR Y)))
                         (T (CAR Y)))
                   (QUOTE SERIAL)))
                 (QUOTE CONSEC))
                Z)))))
           (T (ERROR (PRINTQ ILLEGAL SENTENCE TERMINATOR))))))
EXPR)

(DEFPROP PARSE1
 (LAMBDA(S)
  (PROG  (P LEX Z)
         (SETQ P NIL)
   LOOP  (COND
          ((NULL S)
           (COND
            ((NULL P) (ERROR (PRINTQ SYNTAX ERROR)))
            (T
             (RETURN
              (LIST
               (#MK-DOMINANT (#MK-OCC)
                             P
                             (QUOTE CONSEC)))))))))
         (SETQ LEX (CAR S))
         (COND
          ((MEMQ LEX (QUOTE (#ACCT #STOP)))
           (COND
            ((NULL (CDR S))
             (ERROR
```

```
      (PRINTQ ACCENT
              or
              STOP
              not
              followed
              by
              an
              atom)))
    ((ASSOC (CADR S) SYMS)
     (ERROR
      (PRINTQ ACCENT
              or
              STOP
              followed
              by
              an
              illegal
              character)))
    ((ATOM (CADR S))
     (SETQ
      P
      (KONS
       P
       (#MK-DOMINANT
        (#MK-OCC)
        (LIST (#ADD-FEATURE (#MK-OCC) T LEX)
              (#ADD-FEATURE (#MK-OCC) T (CADR S)))
        (QUOTE CONSEC))))
     (SETQ S (CDDR S))
     (GO LOOP))
    (T
     (ERROR
      (PRINTQ ACCENT
              or
              STOP
              may
              not
              be
              followed
              by
              a
              parenthesized
              expression)))))
  ((MEMQ LEX (QUOTE (#COM #COLN #SEMI #PAUS)))
   (SETQ Z (PARSE1 (CDR S)))
   (COND
    ((ATOM (CAR Z))
     (RETURN
      (CONS
       (LIST
        (#MK-DOMINANT (#MK-OCC) P (QUOTE CONSEC))
        (#ADD-FEATURE (#MK-OCC) T LEX)
        (CAR Z))
       (CDR Z))))
    (T
```

```
                (RETURN
                 (CONS
                  (*APPEND
                   (LIST
                    (#MK-DOMINANT (#MK-OCC) P (QUOTE CONSEC))
                    (#ADD-FEATURE (#MK-OCC) T LEX))
                   (CAR Z))
                  (CDR Z))))))
              ((MEMQ LEX (QUOTE (#EXCL #DOT #QUES)))
               (RETURN
                (CONS (#MK-DOMINANT (#MK-OCC) P (QUOTE CONSEC))
                      S)))
              ((ATOM LEX)
               (SETQ P (KONS P (#ADD-FEATURE (#MK-OCC) T LEX)))
               (SETQ S (CDR S))
               (GO LOOP))
              (T (SETQ Z (PARSE1 LEX))
                 (NILL LEX IS A LIST)
                 (COND
                  ((NULL (CDR Z)) (SETQ P (KONS P (CAR Z)))
                                  (SETQ S (CDR S))
                                  (GO LOOP))
                  (T
                   (ERROR
                    (PRINTQ illegal
                            nesting
                            of
                            expressions)))))))
EXPR)
```

There is not space here, or inclination, to survey all these functions, so we will briefly review just a few of them. The function #MK-OCC generates a new occurrence.

```
(DEFPROP #MK-OCC
 (LAMBDA NIL
  (PROG  (X)
         (SETQ X (INTERN (GENSYM)))
         (PUTPROP X T (QUOTE OCCURRENCE))
         (PUTPROP X (INTERN (GENSYM)) (QUOTE NEUTRAL))
         (SET X X)
         (RETURN X)))
EXPR)
```

The function #MK-DOMINANT does just that - itspecifies that a given list of occurrences is subordinate to another specified occurrence.

```
(DEFPROP #MK-DOMINANT
 (LAMBDA(X Y ORDER)
  (PROG  (A)
         (RETURN
          (COND
           ((GET X (QUOTE OCCURRENCE))
            (MAPCAR
```

```
                (FUNCTION
                 (LAMBDA(Y)
                  (COND
                   ((GET Y (QUOTE OCCURRENCE)) T)
                   ((ERROR
                     (PROGN
                      (PRINT* (LIST Y))
                      (PRINTQ NOT
                              AN
                              OCCURRENCE
                              -
                              #MK-DOMINANT)))))))
                 Y)
                (SETQ A (GET X (QUOTE NEUTRAL)))
                (SELECTQ
                 ORDER
                 ((QUOTE UNORDERED)
                  (PUTPROP A (*APPEND Y (GET A ORDER)) ORDER))
                 ((QUOTE CONSEC)
                  (PUTPROP A (CONS Y (GET A ORDER)) ORDER))
                 ((QUOTE SERIAL)
                  (PUTPROP A (CONS Y (GET A ORDER)) ORDER))
                 ((ERROR
                   (PROGN
                    (PRINT* (LIST ORDER))
                    (PRINTQ INVALID
                            ORDER
                            CLASS
                            -
                            #MK-DOMINANT)))))
                 X)
                ((ERROR
                  (PROGN
                   (PRINT* (LIST X))
                   (PRINTQ NOT
                           AN
                           OCCURRENCE
                           -
                           #MK-DOMINANT)))))))
EXPR)
```

The function FRAME takes an occurrence or list of occurrences and marks the items specified in the second argument list. It calls ADD-FEATURE and #MK-DOMINANT, because those are the two principle functions it specifies.

```
(DEFPROP FRAME
 (LAMBDA(OCC SEQ)
  (SEARCH OCC
          (#MK-DOMINANT (#MK-OCC)
                        (COMPOSE SEQ)
                        (QUOTE UNORDERED))
          (FUNCTION
           (LAMBDA (Y) (#ADD-FEATURE Y T (QUOTE #FRAME))))))
EXPR)
```

```
(DEFPROP #ADD-FEATURE
 (LAMBDA(X SIGN F)
  (COND  ((GET X (QUOTE OCCURRENCE))
          (PUTPROP
           X
           (CONS (CONS F SIGN) (GET X (QUOTE FEATURES)))
           (QUOTE FEATURES))
          X)
         ((ERROR
           (PROGN (PRINT* (LIST X))
                  (PRINTQ NOT
                          AN
                          OCCURRENCE
                          -
                          #ADD-FEATURE))))))
EXPR)
```

Bibliography

[Anderson, 1980] Anderson, J. R., "Arguments concerning representations for mental imagery," in [Seamon, 1980], pp. 243 - 271.

[Brinch Hansen, 1973] Brinch Hansen, P., Operating System Principles, Engelwood Cliffs, N.J.: Prentice-Hall.

[Brown, 1972] Brown, G. Spencer, Laws of Form, New York: The Julian Press.

[Chomsky, 1957] Chomsky, Noam, Syntactic Structures, The Hague: Mouton.

[Chomsky, 1961] "Some Methodological Remarks on Generative Grammar", Word, (17), pp. 219 - 239. Selected parts reprinted in [Fodor, 1964] as "Degrees of Grammaticalness", pp. 384 - 389.

[Chomsky, 1965] Chomsky, Noam, Aspects of the Theory of Syntax, Cambridge: M.I.T. Press.

[Chomsky, 1968] Chomsky, Noam, Language and Mind, New York: Harcourt Brace Jonanovich.

[Chomsky, 1971] Chomsky, Noam, "Deep Structure, Surface Structure, and Semantic Interpretation", in [Steinberg, 1971], pp. 183 - 216.

[Chomsky, 1975] Chomsky, Noam, The Logical Structure of Linguistic Theory, New York: Plenum Press.

[Chomsky, 1977] Chomsky, Noam, Essays on Form and Interpretation, New York: North-Holland.

[Ciba, 1979] CIBA Foundation Symposium 69, Brain and Mind, New York: Excerpta Medica.

[Church, 1936] Church, Alonzo, "A note on the entscheidungsproblem", The Journal of Symbolic Logic, (1:1), pp. 40 - 41.

[Church, 1941] Church, Alonzo, "The Calculi of Lambda Conversion", Annals of Mathematics Studies no. 6, Princeton, N.J.: Princeton University Press.

[Davidson, 1972] Davidson, Donald and Harmon, Gilbert (Eds.), Semantics of Natural Language, Boston: D. Reidel.

[Fillmore, 1968] Fillmore, Charles J., "The Case for Case", in [Bach, 1968], pp. 1 - 88.

[Fillmore, 1972] Fillmore, Charles J., "On Generativity", in

[Peters, 1972], pp. 1 - 19.

[Fillmore, 1971] Fillmore, Charles J. and Langendoen, D. T., Studies in Linguistic Semantics, New York: Holt, Rinehart and Winston.

[Fodor, 1975] Fodor, Jerry A., The Language of Thought, Cambridge: Harvard University Press.

[Fodor, 1964] Fodor, Jerry A. and Katz, Jerrold J., The Structure of Language - Readings in the Philosophy of Language, ENglewood Cliffs, N.J.: Prentice-Hall.

[Granit, 1955] Granit, Ragnor, Receptors and Sensory Perception, New Haven: Yale University Press.

[Gruber, 1976] Gruber, Jeffrey S., Lexical Structures in Syntax and Semantics, New York: North-Holland.

[Jackendoff, 1972] Jackendoff, Ray S., Semantic Interpretation in Generative Grammar, Cambridge: M.I.T. Press.

[Jardine, 1975] Jardine, Nicholas, "Model Theoretic Semantics and Natural Language", in [Keenan, 1975], pp. 219 - 240.

[Katz, 1977] Katz, Jerrold, Propositional Structure and Illocutionary Force, Cambridge: Harvard University Press.

[Katz, 1963] Katz, Jerrold J. and Fodor, Jerry A., "The Structure of a Semantic Theory", in [Fodor, 1964, pp. 479 - 518.

[Keenan, 1975] Keenan, Edward L. (Ed.), Formal Semantics of Natural Language, Cambridge: Cambridge University Press.

[Kleene, 1936] Kleene, Stepehen Cole, "General Recursive Functions of Natural Numbers", Mathematische Annalen, (112), pp. 727 - 742.

[Kosslyn, 1978] Kosslyn, S., "Imagery and Internal Representation," in [Rosch, 1978], pp. 217 - 257.

[Kosslyn, 1980] Kosslyn, S. M., Murphy, G. L., Bemesderfer, M. E. and Feinstein, K. J., "Category and continuum in mental comparisons," in [Seamon, 1980], pp. 207 - 242.

[Kowalski, 1979] Kowalski, Robert, Logic for Problem Solving, New York: North Holland.

[Lakoff, 1967] Lakoff, G. and Ross, J. R., "Is Deep Structure Necessary?" Duplicated, M.I.T.

[Lakoff, 1971] Lakoff, George , "On Generative Semantics", in [Steinberg, 1971], pp. 232 - 296.

[Lakoff, 1972] Lakoff, George, "Linguistics and Natural Logic",

in [Davidson, 1972], pp. 545 - 665.

[Lakoff, 1980] Lakoff, George and Johnson, Mark, Metaphors We Live By, Chicago: The University of Chicago Press.

[Langer, 1942] Langer, Susanne K., Philosophy in a New Key, New York: Mentor Books.

[Mountcastle, 1974] Mountcastle, Vernon B., Medical Physiology Volume One, Saint Louis: The C. V. Mosby Company.

[Perkel, 1969] Perkel, D.H. and Bullock, T.H., "Neural Coding", in Neurosciences Research Symposium Summaries, edited by F.O. Schmitt, T. Melnechuk, G.C. Quarton and G. Adelman, Cambridge: M.I.T. Press.

[Peters, 1972] Peters, Stanley (Ed.), Goals of Linguistic Theory, Englewood Cliffs, N.J.: Prentice-Hall.

[Post, 1943] Post, Emil L., "Formal Reductions of the General Combinatorial Decision Problem", American Journal of Mathematics, (65), pp. 197 - 215.

[Potts, 1975] Potts, Timothy C., "Model Theory and Linguistics", in [Keenan, 1975], pp. 241 - 250.

[Putnam, 1979] Putnam, Hilary, in [Ciba, 1979], p. 366.

[Robinson, 1979] Robinson, J.A., Logic: Form and Function, Edinburgh: Edinburgh University Press.

[Schoenfinkel, 1924] Schoenfinkel, Moses, "On the Building Blocks of Mathematical Logic", in [van Heijenoort, 1967], pp. 355 - 366.

[Scott, 1970] Scott, Dana, Outline of a Mathematical Theory of Computation, Oxford University Computing Labs, PRG-2, Oxford University.

[Seamon, 1980] Seamon, John G., Human Memory - Contemporary Readings, New York: Oxford University Press. [Steinberg, 1971] Steinberg, Danny D. and Jakobovits, Leon A. (Eds.), Semantics: An Interdisciplinary Reader in Philosophy, Linguistics and Psychology, Cambridge: Cambridge University Press.

[Stent, 1978] Stent, G.S., Kristan, W. B., Friesen, W. O., Ort, C. A., Poon, M. and Calabrese, R. L., "Neuronal Generation of the Leech Swimming Movement," Science, (200), pp. 1348 - 1357.

[Turing, 1937] Turing, A. M., "On Computable Numbers with an Application to the Entscheidungsproblem," Proceedings of the London Mathematical Society, ser. 2 (42), pp. 230 - 265; Correction, ibid (43) pp 544 - 546.

[Ullmann, 1959] Ullmann, Stephen, The Principles of Semantics, Oxford: Basil Blackwell.

[van Heijenoort, 1967] van Heijenoort, Jean, From Frege to Goedel - A Source Book in Mathematical Logic, 1879 - 1931, Cambridge: Harvard University Press.

[Weinreich, 1980] Weinreich, Uriel, On Semantics, University of Pennsylvania Press.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


FILMED
8