CUCS-2-77

Structure and Abstraction in a System for Conceptual Matching

David Elliot Shaw

August 1977

Abstract

The notion of a *conceptual matching task* is introduced as a model for a number of important practical problems in which systematically structured entities are paired on the basis of selected significant characteristics of their structures. A particular task paradigm involving the interactive description of matchable entities and their subsequent pairing by computer is specified, along with the essential characteristics of a structured system which distinguish such a task as a *conceptual* matching problem. In particular, the perception of characteristic elements, properties and relationships which emerge in the context of different *structural* and *abstractive viewpoints* is seen to be critical to this sort of task. The representational and organizational framework of a proposed system for conceptual matching is outlined. This system uses a uniform *schematic* formalism for representing both objects and processes, including those processes from which the procedural component of the system itself is built. The operation of our system centers on an *active memory* of limited size which serves as the site of *conceptual integration*, and which is distinguished from the *archival memory* by the availability of *descriptive access mechanisms* for manipulating its contents. The use of these descriptive mechanisms in *elaborating* and *specializing* high-level guiding frameworks, as required in both the descriptive and matching subtasks, is illustrated.

Acknowledgements

The author is especially indebted to Terry Winograd and Ed Parker, of Stanford University, and to Alan Borning, of the Xerox Palo Alto Research Center, for suggesting many of the ideas which have been incorporated in this paper, and for providing much of the feedback which made possible their expression.

STANFORD SYSTEMS CORPORATION

TECHNICAL MEMO SSC-IR-7702-01

Structure and Abstraction in a System for Conceptual Matching

David Elliot Shaw

August 1977

Suite 1020 525 University Avenue Palo Alto, California 94301 (415) 321-8111

Section 1
Introduction to Conceptual Matching

Many information processing tasks performed by men and machines alike involve the process we call *matching*, by which a correspondence is assigned between members of two sets of entities. The criteria for certain sorts of matches are quite simple to describe. Letters are routinely matched with mailboxes, for example, and Congressmen with constituents, according to straightforward algorithms based on simple, single properties of the entities in question. In this paper, though, we will consider certain essential aspects of a more interesting class of tasks which will be referred to as *conceptual* matching problems. This more demanding sort of task, which is nonetheless a common part of our cognitive experience, involves the assignment of a correspondence between entities which humans perceive as being highly and systematically structured, based on selected significant characteristics of those structures.

Our central interest is in the isolation of mechanisms essential to the implementation of computer-based systems for the successful execution of conceptual matching tasks. In Section 2, we thus define a two-phase functional task paradigm involving interactive description-building and subsequent computerized matching for use as a vehicle to explore those issues of knowledge representation and manipulation most critical to the problems of conceptual matching. Numerous information processing tasks of considerable social and economic import may be regarded as instances of our conceptual matching paradigm. One of the most important current examples is the problem of bibliographic search, in which the matchable entities are the current information needs of a reader, on the one hand, and individual publications, on the other. Although information retrieval is currently enjoying both extensive attention in research and widespread practical application, an incapacity for *understanding the content* of the indexed materials imposes a critical limitation on the utility of computer-based document retrieval.

The present limitations of manual and computer-based systems for bibliographic search stand in evidence of the apparent difficulty of explicitly modeling this sort of understanding. While we will not attempt a systematic review of the performance of the various document retrieval schemes currently in use, an anecdotal illustration of the pitfalls of one superficially plausible method for locating relevant published material is provided by the problems experienced by this author in a recent manual literature search. Using a *permuted keyword* index, in which the corpus of publications was indexed by combinations of significant words appearing in the title, we attempted to locate all articles dealing with the application of computers to job placement, and with those aspects of "job analysis" which might be relevant to the representation of occupational information in a computer.

While it is difficult to estimate the number of *relevant* articles which were *not* retrieved using the keyword method, nearly all of the papers which *were* located using this scheme were found, on examination of their full titles, to be irrelevant to the intended focus of the search. One promising prospect indexed with the key words *job, analysis, computer* and *system* led to the full title "Preliminary Job Analysis of Computer Programmer and Systems Analyst", in which the conceptual elements of our intended target class are combined to indicate a very different subject area. Even more divergent from the search goal was a paper entitled "Employment of a Computer System for Acquisition and Processing of Measuring Data in Nuclear Physics Experiments in ZFK Rossendorf", located through the conjunction of the keywords *computer, system,* and *employment*. Our most cherished find, though, was a second publication located using the terms *job* and *analysis*: "A Behavioristic Analysis of the Book of Job".

These examples, of course, are not advanced as a serious argument against the adequacy of all existing bibliographic search mechanisms. In the context of a discussion of the nature of human matching in the process of document retrieval, though, it is interesting to observe that even *biblical* knowledge may useful in eliminating candidates in the course of a search for articles discussing the application of computers to job placement. Clearly, the effective retrieval of user-relevant literature from among a large body of published material constitutes a matching task which would intuitively be considered conceptual in nature. The characteristics of this sort of matching process, whether manifested in a human or computational host, will form a central focus of this paper.

A second important instance of the conceptual matching paradigm discussed in the current paper is suggested by the intended target of our permuted keyword search: the problem of finding employment for a large set of job-seekers. The appearance of numerous human economic agents in both the public and private sectors who specialize in just this matching task suggests that the process is a nontrivial one which may require the application of the human conceptual abilities in which we are interested. One might conceive of a very straightforward method by which the skills of each job-seeker, along with the requirements of each available position, would be classified according to a fixed, common classificatory scheme and the two sets compared to yield lists of exact matches. Unhappily, though, experience has shown such procedures to yield a large number of "unsatisfactory" pairings, and to fail to discover many "appropriate" matches, by comparison with a human employment agent.

To gain some feeling for the complexity of the job-matching task, one need only examine the kinds of judgements made by a skilled employment agent or personnel director in the course of his work. In contrast to the simple scheme just outlined, an effective human job-matcher can not deal with a job as if it were an indistinguishable member of a single occupational category, but must make use of various characteristics of the position in the context of different potential matches. In the course of finding a new position for an engineer having extensive sales experience, for example, a different set of occupational skills may become salient depending on the potential positions for which he is being considered. In assessing his suitability for a position as marketing manager, we may wish to regard the job-seeker as a former salesman; for the position of director of research and development, sales experience may still be relevant, but we are likely to think of his previous job primarily as a technical position.

Indeed, the single-category matching model might be extended to allow our engineer membership in two or more occupational classes so that both his technical *and* marketing skills might be considered in effecting his placement. In actual experience, though, even such multiple classification schemes fail to capture enough of the meaning of the jobs they describe to emulate the performance of a human matcher. Let us, for example, construct two hypothetical positions which might both be classified as requiring experience in the areas of "air pollution" and "electronics". The first involves the development of electronic devices for monitoring air pollutants; the second deals with the control of emitted byproducts in the semiconductor manufacturing industry. Clearly, the job skills salient to the performance of these two jobs are quite different, and we would expect any competent occupational counselor to distinguish between the sorts of experience relevant to each. The human employment agent does not regard the required background for a position as an unstructured list of experience categories, but can perceive the essential *relationships* between several areas of prerequisite experience within the context of a functional understanding of the particular job.

Another characteristic feature of conceptual matching processes is the ease with which novel matching situations are accomodated through the appropriate application of experience in *related* areas. In 1970, for example, a job such as microprocessor systems assembly line manager did not exist. With the advent of microcomputer technology, the data processing employment agent did not simply assign a new and unrelated conceptual pidgeon-hole to this emerging position, but instead was able to make use of his previous knowledge of a number of related occupational areas to distinguish salient criteria for matching workers to jobs involving microprocessors.

A surprising degree of commonality is evident among the critical problems involved in diverse applications of conceptual matching. The analysis of these problems which is outlined in Section 2 is based on an examination of the distinctive qualities of those entities which must be matched in a "conceptual" manner, and of the criteria upon which the appropriateness of such matches is evaluated. In particular, the notions of *structural* and *abstractive* levels will be introduced as dimensions for characterizing the perspective from which a structured entity may be viewed in the context of a matching task. These two constructs will then be used to formulate a model of conceptual matching based on the iterative *specialization* and *elaboration* of conceptual structures. In Section 3, the central mechanisms of this model are incorporated in a proposed organizational framework for schematic, description-based system which builds and matches conceptual descriptions.

Section 2
The Conceptual Matching Problem

In this section, we will define and examine the central problems of a class of conceptual matching tasks which might be executed by a man-machine system involving one or more human users and an "intelligent" computer-based subsystem. In the first subsection, the mechanics of the chosen task paradigm are outlined in a straightforward functional problem specification. Those special properties of the matchable entities and of the criteria for justifying derived pairings which distinguish our task as a problem in *conceptual* matching are then introduced in Subsection 2.2. The section ends with a discussion of the description and matching subtasks in light of these special properties.

2.1 Functional Task Specification

The conceptual matching task to be realized by our proposed thesis system involves the pairing by computer of members of two distinct *match sets* of structured entities on the basis of machine-readable descriptions of those entities previously constructed in an interactive manner. We may thus divide the task into two temporally independent subtasks, the first involving the interactive *description* of the members of each set, and the second consisting of the actual *matching* process. (In the remainder of this proposal, we will generally use the term "matching" to refer specifically to the latter of these two subtasks. Use of the term in reference to the conceptual matching task as a whole should be obvious from context.) We are thus interested in the problem of matching entities based on *static* machine-readable descriptions -- symbolic representations of significant aspects of the elements of one match set which are prepared before the identity of those elements of the other set with which they will be matched is known.

The possible interactive modes which might be considered as candidate frameworks for the description-building process may be characterized according to their position on a spectrum extending from machine-guided to user-guided interaction. Toward the user-guided end of this spectrum would be a system allowing unrestricted description of the significant characteristics of an entity by the user in a natural language such as English. The computer-guided end of the spectrum might be exemplified by a mode in which a description is incrementally formulated based on the answers to a fixed sequence of multiple-choice questions presented to the user. On the one hand, we would like a conceptual matching system to accept input felt by the user to express distinctive characteristics in a direct manner, thus obviating the need for numerous queries from the system which may be inapplicable to the entity in question. At the same time, it will generally be important to allow the machine to guide the description-building process toward information relevant to anticipated matching sessions on the basis of a conceptual model of the subject domain, and to solicit such amplifying or clarifying information as may be useful.

The particular description-building paradigm chosen for our current investigations involves the solicitation of user responses in a tightly constrained form and a dynamic focusing of the query sequence on the basis of these responses. Specifically, the selection of multiple-choice items from a fixed "menu", or the provision of single-word or fixed-phrase responses recognized directly by the system are input mechanisms consistent with our description-building paradigm. In order to simplify the "front end" of our system, allowing particular focus on our central research interests, we will not attempt the interpretation of unrestricted natural language responses or expressions in a formal language of sufficient expressive power that involved parsing or pattern-matching techniques are required for their assimilation. Like a human participant in a game of "twenty questions", though, the incremental description-building process will use previous responses to direct further queries in such a way that the system converges rapidly on the essential characteristics of an entity. Our paradigm thus provides for the integration of user-provided and system-provided guidance in an interactive description-building process.

Let us now consider the functional framework of the matching subtask itself. In specifying the external behavior of the matching process, one of the two match sets is distinguished as the *selective* set, while the other is termed *selectable*. For each member of the selective set (called *selective entities*), the matching process returns a list of *selected entities* from the selectable set which match the given selective entity according to criteria which we will discuss shortly. To be useful, of course, this list should contain significantly fewer members than the selectable set itself. Because our primary concern is with the conceptual mechanisms involved in the match process, our matching system will also provide an explanation in some form of the basis on which the selective entity was paired with each selected entity in the output list. The form and significance of these symbolic *match justifications* will be clearer following our discussion of the conceptual aspects of the match process. Upon completion of the pairing process for each entity in the selective set, the matching process is terminated, yielding a list of plausible pairings along with the conceptual arguments used to justify them. In a realistic application, the output list could be reviewed by a human agent, who might prune the set of selected entities using judgemental mechanisms still unattainable in a computer-based system.

The criteria used to evaluate the appropriateness of a potential pairing are best understood in the context of the concerns of Subsections 2.2 and 2.3. For now, it should be mentioned that these criteria may highly specific to the particular selective or selectable entity under consideration, or to the interaction of properties of each. Thus, it may in general be difficult to make explicit in the match criteria applied to a particular matching task apart from the context of a specific candidate pair. In one extreme case which is nonetheless consistent with the constraints of our general matching paradigm, the criteria for matching each selective entity with an appropriate partner may be ideosyncratic tests having little in common with those corresponding to other elements of the selective set. The match criteria for a particular task may also be closely tied to individual *selectable* entities, to the possible relationships *between* selective and selectable entities, to very general considerations applied in a more or less uniform way to all matchable entities, or to specific areas of detailed world knowledge embedded in the matching system but not explicit the descriptions of the matchable entities, Thus, we must view the notion of a criterion for the evaluation of match appropriateness in exceedingly general terms.

2.2 Essential Characteristics of Structured Systems

The most interesting aspects of conceptual matching are related to the fact that both the matchable entities and the match criteria exhibit a rich conceptual structure. It is the perception of this conceptual structuring which makes possible the sort of guidance offered by the system in the process of description-building, and which provides the conceptual basis for evaluating the appropriateness of potential pairings in the match process itself. In this subsection, we will examine those distinguishing aspects of structured systems which lend a conceptual flavor to the class of matching tasks with which we are concerned. In the course of this discussion, we will introduce the notion of *structural level*, which reflects the *scope* and *grain* of a particular observation of some structured entity. The range of possible levels of perceived detail will be conceived of as a *structural hierarchy* (related to constructs discussed by Simon [196<n>] and in a collection edited by Pattee [197<n>]) in which a distinctive set of *emergent properties* are made evident at each higher level through the imposition of *constraints* on the configuration of lower-level elements. We will then discuss the notion of *structured abstraction*, by which *sets* of systems are perceived as systematically related in different ways according to the nature and restrictiveness of these constraints. The properties of structural constraints will be used in Subsection 2.3 to interrelate the characteristics of structural detail with the principles of abstraction in a manner which will be useful in examining the important problems of conceptual description-building and matching.

Loosely speaking, a system may be considered structured whenever it can be fruitfully examined at more than one level of conceptual detail, with different properties of the system becoming salient to the observer depending on the chosen "grain" of observation. An automobile company, for example, may be observed at any of several levels of detail, each associated with different sorts of characteristics. Viewed at a detailed structural level, the activities of each automotive worker may be described in terms of such basic physical activities as hand and arm movements, the production of phonemic strings, and primitive acts of personal locomotion. With the addition of a detailed mechanical description of each machine found in each manufacturing plant, we might imagine the possibility of providing a very *accurate* description of the detailed functioning of the firm over some period of time. Such a description, though, would have little value in coherently expressing those aspects of these operations which would be considered most important to a productive business enterprise. While a detailed analysis at this level would be essential for the solution of certain problems, it is unlikely that a consumer comparing the advantages of various makes and models, for example, or an investor considering the purchase of stock in the company, would be interested in considering the detailed behavior of each machine and employee in the firm.

Ascending the structural hierarchy to a level of detail corresponding to the corporation as a whole, though, a very different set of characteristics *emerges* from the same physical entity. At this new structural level, we might observe such features as a product, a board of directors, and such abstract notions as capital assets. Each of these is in fact made up of components observed at lower levels, but the observation of *characteristic* elements and inter-element relationships makes it possible to view these same phenomena at a "higher" structural level which is likely to be of greater use to an investor. A historical analysis of industrial production during the Second World War or an investigation of current unemployment patterns would probably involve a still higher structural level corresponding to the automobile *industry* taken as an aggregated whole. The level in the structural hierarchy which is chosen for observation is thus highly dependent on the particular purpose of that observation.

It is significant that not *all* possible interacting combinations of people, equipment and actions can be usefully observed in terms of the characteristic features of automobile firms. Rather, it is the introduction of a characteristic set of *constraints* on the detailed configuration of this system which leads to the emergence of conceptually useful properties at the corporate level of the structural hierarchy. Most important, the perception of "similarly" constrained systems may allow an observer to profitably apply a common interpretive framework to related *sets* of configurations. When our potential investor asks his broker about the management of a particular automobile firm, for example, the broker need not explain the function of a corporate president in terms of low-level human behavioral primitives. Perceived commonalities in the sorts of constraints observed in *all* corporations allow the broker to assume that his client is generally familiar with such emergent elements as the general role of a corporate president.

Emergent properties, then, may be associated with similarly constrained *sets* of lower-level configurations. Clearly, though, a given configuration may be regarded as a member of any number of conceivable configuration sets depending on the particular constraints chosen for use in the partitioning process. The set of properties which emerges with a transition to a higher structural level is thus dependent on the particular set of configurations whose characteristic constraints are considered "similar" in the context of current requirements. One degree of freedom is related to the *size* of the set of conceivable systems which are grouped together for purposes of factoring out common emergent properties. At a given structural level, certain emergent characteristics may be shared by a large number of particular detailed configurations, while others may be applicable only to some proper subset of these configurations.

Some of the characteristics of our example system which emerge at the level of the automobile firm, for instance, are in fact features of organizations in general. (Most organizations can be assumed to have members, for example, along with some form of leadership and a purpose for existence). Other characteristics which emerge at the same structural level are particular only to corporations -- the existence of a stockholders, products and a board of directors, for example. When the same system is viewed as a member of a another set of configurations whose detailed components are even more tightly constrained, a third set of properties, relevant only to automobile corporations, emerges. Furthermore, different properties may emerge at this structural level depending on the particular subset which is chosen. A political party, for example, may, like a corporation, be viewed as a specific kind of organization. While lacking stockholders, products and directors, this specialized type of organization reveals such emergent properties as candidates, conventions and Central Committees.

Furthermore, most configurations may be usefully regarded as members of several non-nested configuration sets, each associated with its own set of emergent characteristics. Beer, for example, may be viewed as a special case of either a beverage or an intoxicant, depending on the characteristics salient to the particular conceptual task at hand. Not all beverages qualify as intoxicants, nor are all intoxicants ingested in liquid form, but a number of the salient properties of beer are in fact general characteristics of beverages or intoxicants in general. The salience of alternative sets of constraints thus introduces further possibilities for the choice of properties to be considered emergent at a given structural level.

We have seen that the particular set of characteristics which emerge from a different observational viewpoint is dependent not only on the level of detail chosen for observation, but on the set of "similar" configurations of which the observed system is considered a member. It may thus be useful to consider a second parameter, distinct from the choice of a structural level, as a determinant of the particular emergent interpretive framework which is salient in a given situation. This parameter, reflecting the *extent* and *dimension* of structural constraint, and influencing the specificity and selection of emergent properties, is the basis of the notion of *abstraction*. As in the case of structural detail, it will sometimes be meaningful to speak of *abstractive levels* ranging from specialized to general, each denoting a progressively less restrictive set of structural constraints. It should be noted, though, that the property of abstractive level, unlike that of structural level, imposes only partial ordering on the possible interpretations of a given system -- only those emergent frameworks which correspond to nested configuration sets may be compared as more or less abstract. The incidence of multiple category membership thus precludes the use of a strict tree-structured interpretation of abstractive relationships, suggesting instead the model of a more general non-cyclic graph.

As we shall see in Section 3, the "factoring" of emergent characteristics at appropriate *abstractive* as well as *structural* levels is critical to the organization of our proposed system for computer-based conceptual matching. Part of the significance of the notion of abstractive levels in naturally and artificially intelligent systems is based on on the *inheritance* by specialized configuration sets of the emergent properties of a more general abstractive "parent". Certain properties of an abstractive "child" are specific to the more tightly constained system, and are unrelated to any properties of the parent. The notion of a product, for example, is meaningful in the context of a corporation, but has no analogue in the more abstract case of a general organization. The officers and directors of a corporation, on the other hand, may be viewed as specializations of the notion of leadership which we listed as one of the important features of organizations in general.

A detailed characterization of the relationship between a parent and child in the abstraction hierarchy must thus often include information about abstractive connections between the essential elements of each, often involving the addition of elaborating details or even special exceptions to the general information associated with the parent. It is this sort of relationship *between the structures* of configurations viewed at various abstractive levels which is reflected in models of *structured abstraction* [Brian Smith, personal communication]. In our later discussion of the possible approaches to the implemention of a computer system capable of conceptual manipulation of the emergent properties of structured systems, structured connections between abstractive "siblings" (whose configuration sets intersect, but fail to nest) will be found useful as well. This extended form of structured correspondence is related to the notion of mapping [Moore and Newell, 1973] and to mechanisms proposed as components of "frame systems" [Minsky, 1975].

2.3 Description and Matching of Structured Entities

For pedagogical reasons, we have chosen to introduce the notions of structural and abstractive levels separately, emphasizing the distinction between the choice of a level of *detail* and that of a level of *generality* from which to view a given structured system. In discussing the construction a conceptually useful description of a structured matchable entity, though, or the matching of such entities according to criterial systems which may themselves be regarded as highly structured, it will be useful to consider certain interesting relationships between the chosen level of abstraction and the accessible level of structural detail. It is the notion of structural constraint which links the roles of abstraction and detail in the conceptual matching task.

Within a single structural level, we have noted that both general and specific characteristics may be salient to a given matching task. The more restrictive constraints associated with relatively specialized interpretive frameworks, though, may have a distinctive role in understanding: the provision of a conceptual bridge between structural levels. Notice, for example, that while certain features of an automobile firm are associated with all organizations, most of these features are expressed at a fairly high level of structural detail. There is very little which can be said at the abstractive level common to all organizations which specifies the detailed activities of the workers, the machinery they are likely to use, the products they produce, etc. In the context of that abstractive level corresponding to automobile firms in particular, though, information as detailed as the physical activities of test drivers and the properties of welding arcs may become meaningful.

In general, the range of structural levels at which observation is meaningful deepens when a more restrictive set of constraints is known to characterize the entity being observed. The knowledge that the corporation in question is more specifically an automobile firm allows meaningful elaboration at the level of the particular models of cars produced. Further specification to a particular manufacturer introduces a framework for examination of even a detailed mechanical subassembly characteristic of that make of automobile. It should be noted that the imposition of restrictive constraints highlights a very *select* set of low-level details which become meaningful within a higher-level framework. The abstractive refinement of a high-level framework thus allows the conceptual integration of *significant* low-level details. The high-level organizing framework can be seen as a sort of "cognitive anchor" though which a number of lower-level features may be accessed, each of which may in turn anchor a collection of still lower-level details.

Our conceptual matching task, like most interesting cognitive tasks, will in general require consideration of characteristics observable from a variety of abstractive and structural viewpoints. The example presented in Section 1 of an engineer with sales experience illustrates the salience of different abstractive perspectives depending on the match criteria and on particular selectable entities under consideration. The level of structural detail at which such a pairing must be justified is also dependent on the particular pairing under consideration. For certain available positions, such high-level constructs as "marketing experience" may be salient, while in other cases, detailed information about products sold in previous jobs may be necessary in evaluating the candi
pairing. Our analysis of both the description-building and matching subprocesses i

integration of general and specialized information from multiple structural levels into an interconnected "conceptual web" of structural elements and relationships. This hypothetical web is unified by structured connections among the various abstractive views corresponding to a *given* structural level, on the one hand, and on the other by connections *between* structural levels which are tied largely to the more specific abstractive viewpoints within each structural level. Thus, the relationship between abstractive and structural levels of interpretation is at the core of our model of conceptual matching.

At the beginning of a description-building subtask, only a very general set of high-level structural characteristics would typically be assumed by the machine to be consistent with the entity to be described. In particular, the system can assume the applicability of an abstract conceptual framework common to the description of *any* element which might be expected as a member of the matchable set. The expectations embodied in this framework would be used by the machine to guide the interrogative process toward details useful in verifying and elaborating the general set of assumptions. While the choice of queries in our description-building paradigm is determined by the machine, though, the acceptance of independent, low-level information, possibly not of immediate use in specializing the high-level framework but ultimately essential to the construction of an integrated description, is one of the properties which distinguishes our conceptual system from a simple categorical descriptive scheme. In a system accepting descriptive input in a relatively free format, as constrained by our general paradigm, (e.g, user-provided phrases), low-level details might include unexpected words or word combinations. In a more tightly machine-guided mode of interaction, responses elicited from the user in the course of a directed inquiry could later be utilized for *different* with much the same effect as in the case where these details were actively provided by the user in a freer description mode.

In our model,.the detection of particular elements and relationships in the user responses, whether or not they have already been integrated into the unifying top-level structure, may serve as clues to the presence of a set of constraints characterizing some conceptual framework. If the role of these elements within the top-level framework is already clear, characteristic occurrences, co-occurrences and interrelationships of such elements may indicate the applicability of a more *specific* unifying framework. In the case of as-yet-unintegrated information, the recognition of characteristic relationships may *suggest* a basis for organizing these details within either the top-level structure or a framework of intermediate structural level. Details from the lower structural levels, then, may serve to suggest new higher-level organizing structures or to specialize structures already involved in the emerging description. These new and specialized structural frameworks, of course, may then be employed to guide the elicitation of further details from the user, lending an iterative quality to the process of description-building (see Norman and Bobrow [1975]). Description-building thus involves an interaction between the elicitation and integration of low-level details under the guidance of high-level expectation frameworks and the postulation of new or specialized high-level frameworks on the basis of observed low-level characteristics. These functions are the basis for the processes of *elaboration* and *specialization*, which are central to the system proposed in Section 3. Ideally, the ongoing collection of relevant details and the progressive tightening of perceived constraints at each successively lower structural level (possibly in more than one abstractive dimension) should allow the construction of a unified inter-level conceptual description embodying a great deal of information of relevant to anticipated matching problems.

The matching subtask itself may also be viewed as an iterative process involving "top-down" guidance toward the elaborative details and the "bottom-up" postulation and specialization of unifying frameworks. Initially, the only organizing structure available to the system is a very abstract, high-level framework expressive of the match criteria and any general expectations as to the form of the matchable entities. The low-level details available to the matching process are derived from the distinctive characteristics of the particular matchable entities. As in the case of description-building, the goal of the matching process is the integration of these low-level details within an interconnected network of systemic structures, anchored at the highest structural level by one or more specialized versions of the original top-level framework.

In the case of the matching subtask, though, the top-level framework must be specialized in such abstractive dimensions as will indicate, in conjunction with its hierarchically interconnected web of details, the *appropriateness* of the match, along with a justification for or argument against the candidate pairing. In the domain of information retrieval, for example, the matching of a selective description representing "books dealing with kidnaping" with a selectable description respresenting "a nonfiction book by Steven Weed" might specialize a general document-matching framework to yield an integrated network which justified the pairing as a presumed example of a book written about a newsworthy acquaintence of the author, whose fame was in this case based on her abduction. The processes of specialization and elaboration are thus central to the matching subtask as well as to the construction of the descriptions which are matched. In the next section, we will outline a particular proposed system for conceptual matching which is based on these processes.

Section 3
A System for Conceptual Matching

In this section, we will describe the organization and central mechanisms of a proposed system for conceptual matching. Our system is based on a uniform *schematic* formalism, introduced in Subsection 3.1, for representing both objects and processes, including those processes from which the procedural part of the system itself is built. Our schematic system base is closely related to the proposed elements of "frame systems" [Minsky, 1975], and in particular to the language KRL [Bobrow and Winograd, 1976], but embodies certain design decisions which reflect substantive additional assumptions regarding the important requirements of the conceptual matching task. These distinguishing aspects of the system base are central to the facilitation of flexible mechanisms for *conceptual integration*, on which the description-building and matching mechanisms are based. In Subsection 3.2, the structure and function of these mechanisms is exemplified in the context of a description-building problem from the job placement domain.

3.1 The System Base

The automation of a specific conceptual matching task within our proposed system framework involves the incremental construction by the system architect of modular conceptual schemata representing various objects and processes related to the matchable entities, match criteria, and particular task domain, using the elements of a uniform, domain-independent "system base". The implementor of a system for the placement of computer programmers, for example, might compose a set of schemata representing such items as languages, machines, and perhaps special qualification tests relevant to certain kinds of programming positions. The system base includes a description-based schematic language for the representation of structural and abstractive relationships and a procedural framework for their manipulation. Consistent with our primary intent, this paper will not describe a detailed and comprehensive system base rich enough in primitive operations and data objects to constitute a powerful tool in the hands of an actual implementor. Instead, we will present a parsimonious outline of the essential features of a system base, including a small set of basic declarative forms and a general outline of the proposed procedural organization, which should be adequate for describing the significant aspects of our proposed conceptual matching system.

The declarative component of this hypothetical system base is built from forms which may be regarded as simplified versions of KRL data types, and which are described here using some of the terminology and an adapted version of the syntax of a subset of KRL-0 [Bobrow and Winograd, 1976]. The basic declarative form in our "knowledge base" is a *frame*-like [Minsky, 1975] structure which we call a *schema*. (Similar structures have been proposed by Bobrow [1975], Bobrow and Norman [1975], Schank [1975], and Shaw [1975, 1975(a)].) A schema may be regarded as the representation of a structured system (an object, process, relationship, abstract concept, etc.) regarded at a given structural level and from a given abstractive viewpoint, and corresponds generally to a KRL *unit* whose "descriptive" and "meta-descriptive" elements are not "syntactically" distinguished. (This distinction, while it raises several interesting questions, will not be the discussed in the current paper.)

Formally, a schema is a defined as a collection of named *slots*. In each schema, one slot, distinguished with the name SELF, serves to describe the entity represented by the schema, possibly from more than one viewpoint. The remaining slots, each named uniquely within the given schema, describe *partial* restrictions associated with *characteristic elements* which emerge when the referent entity is viewed at the given structural and abstractive level. Each slot, including the SELF slot, is a named description expressing certain partial constraints, which may have the form of *relational* constraints involving other slots. Slots may in general be made up of more than one *descriptor*, each characterizing its potential filler in one of three different ways.

The first form, called an *abstractive* descriptor, characterizes its referent as a member of some more general set of structured configurations, optionally imposing supplementary structural constraints on the characteristic elements of this abstractive parent. The second form is called a *component* descriptor, and describes its referent as a characteristic element of some other entity, viewed from a particular structural and abstractive viewpoint. The final form of descriptor is a *direct reference* to some particular structured system from an explicitly known viewpoint. These descriptive forms correspond to the KRL-0 perspective, specification and direct pointer, respectively, and have been renamed only to suggest their abstractive and structural roles within our model and to avoid a misleading possible interpretation of the term "specification". The syntax of the three descriptor types, based on those of the corresponding descriptive forms in KRL-0, is specified in Figure 3.1 as part of a syntactic definition of the simple declarative base which we will use in our exposition. Lower case is used to indicate syntactic classes, while upper case words are elements of the descriptive language itself. Arrows, vertical bars, square brackets and asterisks are meta-linguistic symbols indicating class definition, disjunctive choice and repetition of zero or more instances, while parentheses, braces, angle brackets and equal signs are actual linguistic symbols.

We have indicated that schemata may be used to describe the structure of *processes* along with such "static" structured entities as objects, relations and abstract concepts. While such schemata may be quite useful in representing "external" processes, such as the mounting of an automobile tire, one of their most important applications is in describing the procedural component of the system itself. Each of the numerous procedures from which our proposed system is built is in fact represented by a schematic description, and can be manipulated by the system as a data object. The content of these "internal" process descriptions is closely related to the executive mechanisms of the system, and will thus be discussed after our proposed functional system organization has been outlined.

The operation of our system is centered around an *active memory*, which at any given time holds pointers to a limited number of *tokens*. (The capacity of this active memory is a fixed parameter of the system, but may be envisioned as a small constant, perhaps on the order of ten.) The knowledge base of our system consists of an *archival memory* containing an unlimited number of schemata, which is distinguished from the active memory principally by the more restricted methods available for accessing and manipulating its contents. A token represents a "conceptual instance" of one or more *template* schemata in the archival memory, and may regarded as a schema which has no slots of its own, deriving its distinctive characteristics instead from the further restriction of slots inherited from its various templates and the specification of its interrelationships with other tokens. Specifically, a token may be described as a degenerate case of the schematic form introduced above whose properties are manifested entirely in its SELF description. Such a description may include abstractive descriptors characterizing the token as an instance of various templates (although in a somewhat different sense than in the case of an archival schema), possibly constrained by additional slot restrictions, along with component descriptors describing the schema as a slot-filler from some other token. A syntactic description of the token data type is included in Figure 3.1.

Active memory is most significant as the site of the conceptual integration process. Briefly, this process involves the specialization of certain tokens to the point where their descriptors include direct references to other tokens, thus "linking" conceptual instances of high-level and low-level schemata. In the following subsection, we will examine the application of this process to those aspects of the conceptual matching task introduced in Subsection 2.3. An important *internal* application of this process which will be discussed shortly involves the integration in active memory of *process* tokens (those having procedural schemata as their templates) which represent executable system

```
schema →
      schema-name
            <SELF description>
           [<slot-name description>]*

token →
      token-name descriptor*

description →
      descriptor
    | { [descriptor]* }

descriptor →
      (A schema-name WITH
            [slot-name = description]*)
    | (THE slot-name FROM description VIEWED-AS schema-name)
    | schema-name

slot-name → identifier

schema-name → identifier
```

Figure 3.1
Syntax of the Declarative System Base

procedures. An examination of the latter application will illustrate one of the distinctive aspects of our proposed system design: a close relationship between the mechanisms for "focusing attention" and for scheduling the execution of multiple *executable* tokens.

As we have suggested, the ultimate distinction between archival and active memory is based on the different mechanisms by which their contents may be accessed. Specifically, independent access to archived schemata is restricted to *direct reference*, requiring the availability of an actual pointer to the schema being referenced, or equivalently, of a globally unique schema name which allows its retrieval without *searching* any part of the archival memory. Tokens, while never resident in the archival memory, may be accessed *by description* whenever they are either resident in active memory or linked in a describable relationship to some other active token. Tokens accessible in either way are termed *descriptively accessible* through active memory, as are the templates of descriptively accessible tokens and any schemata which are descriptively linked to these templates. Certain simple active memory references (those involving the retrieval of any active token having some specified template, for example, or filling a given slot in an active token having some specified template) will involve the straightforward execution of system primitives. As we shall see, though, access to descriptively accessible tokens and schemata may at the other extreme involve the execution of an integrated routine built from highly specialized procedural tokens which were *themselves* assembled in active memory. In either case, descriptive access capabilities provide for the retrieval of tokens on the basis of partial specifications which are interpreted in an *active context* expressing the current "attentive focus" of the system, and for selective access to the templates and other archival schemata on which the characteristics of these tokens are based.

All actual activity within our system is directly or indirectly initiated by an *executive* capable of interpreting a restricted set of *system primitives*. Code built from these primitives is associated with a subset of the internal process schemata stored in archival memory, and may reference certain slots of these schemata which serve the functions of formal parameters, including procedural parameters which may be bound to appropriate process tokens. A token which has a code-bearing schema as one of its templates is distinguished as *executable*, and may, through the imposition of supplementary slot restrictions on the code-bearing template, be used to bind these parameter slots for execution. Following the possible binding of some (though not necessarily all) of the parameter slots, one of several mechanisms, described shortly, may cause the *execution* of a descriptively accessible executable token using the code of a specified one of its templates and the (often incomplete) set of bindings. While it will not be possible in the current paper to detail a full set of system primitives by which executable tokens ulimately manifest their effects, it would be well to consider at this point the basic categories of executable primitives necessary to support the sort of system we are describing.

Our proposed system, like most typed symbolic programming languages, includes *type-specific* primitives for the *construction*, *access*, and *comparison* of its data structures. Schematic access primitives include an operation which returns an archived schema specified by a direct pointer or unique name. The system also includes primitives for returning and successively selecting *all* tokens currently resident in active memory. As suggested earlier, though, most contextual access processes of interest involve more complex routines built from the system primitives, often through the specialization of high-level tokens representing conceptual instances of general access *frameworks* and their elaboration using lower-level procedural tokens. Access primitives which return the description associated with a particular named slot of some given schema are also included in the system base. Other primitives are provided for enumerating the constituent descriptors of a given description. Type-specific comparison primitives allow the executive to perform simple equality comparisons between basic data elements of identical type. As in the case of access, though, the most "visible" comparison processes typically involve correspondences between systemic structures, which may introduce the full complexity of a subsidiary conceptual matching task. Finally, the system includes primitives for constructing the two types of descriptors, for constructing and amplifying named descriptions using these descriptors, and for assembling these descriptions to form tokens and schemata.

In addition to these type-specific mechanisms for manipulating elements of the declarative base, the system provides a set of basic *control* primitives, such as might be found in a typical LISP system, along with standard operations for *input* and *output*. Two additional classes of functions, particular to the special characteristics of our system, are included as primitives. First, the system provides primitives for the *activation* and *deactivation* of (procedural and non-procedural) tokens. Second, an *execution* primitive, applicable only to an executable token, is included to allow interpretation of the direct code associated with one of its code-bearing templates. This primitive specifies not only a particular descriptively accessible token, but a specific member of the set of code-bearing templates whose code is to be executed, thus permitting the association of more than one routine with the same executable token. As indicated earlier, the evaluation of code associated with a template of an executable token may be initiated in several ways. One method involves the issuance of an explicit command by the user, typically at the point of task initiation. Most of the actual processing in our proposed system, though, is generally carried out by tokens whose execution has been initiated through the evaluation of execution primitives from within *other* currently executing tokens.

Unlike the execution of an ordinary LISP program, our system supports the (conceptually, if not physically) *concurrent* execution of more than one token under a possibly heterarchical [Minsky, 1973] control regime. In the case where one executing token activates, and subsequently initiates execution of, a second executable token, the system base imposes no fixed protocol for the return of either descriptive "results", residence in active memory, or executive control to the calling token. Applications such as the common use of a token which is being jointly specialized and elaborated by two co-routines, or the interpolation of special error-handling routines to handle exceptional conditions, are supported by this non-hierarchical control discipline.

Among the user-initiated executable tokens is generally a high-level "supervisor" (distinct from the system executive, which is capable of evaluating only "directly executable" code built from system primitives) which oversees this execution process on the basis of information associated with various executable and non-executable tokens currently accessible by description. Among the typical functions of such a supervisor are the allocation of processing and active memory resources (using the execution, activation, and deallocation primitives) and the activation of procedural and non-procedural tokens relevant to the solution of such processing irregularities as looping, deadlocks and recognizable system "bugs". Structural information relevant to such a supervisory process may be associated with the executable tokens themselves, with their code-bearing and non-code-bearing templates, and with various procedural and non-procedural tokens with which they are descriptively linked. The user-initiated supervisor (or, as we shall see, any other independent or subsidiary executable token) may thus serve as a sort of *meta-processor*, capable of examining the structural and abstractive properties of executable, and even executing, processes and altering the current state of the system accordingly. Such alterations may be effected by modifying either the set of tokens which are descriptively accessible (the active tokens, along with those tokens which are descriptively linked to some active token), the "agenda" of active tokens which are currently executing or awaiting execution, or the contents of the descriptively accessible tokens themselves.

With the exception of the special relationship between a user-initiated supervisor and the very general class of tokens whose execution it may initiate, we have thus far spoken of the executable component of the system base as if it were a collection of *indivisable* routines associated with the templates of executable tokens. As we have suggested, though, the schematic part of these templates (by contrast with their associated directly executable code) may in general include slots which describe other internal processes. One (although not the only) important function of this sort of procedural slot is the decomposition of a process into a number of constituent subprocesses which, unlike an ordinary subroutine, may be identified by description rather than direct reference. Other elements of the invocation of a constituent procedure, such as the identity of the "parameters" with which the process is to be called, may also be partially rather than totally constrained in the process template.

This capability allows the construction of routines composed of a number of *descriptively linked* procedures, which may consequently be quite general, calling (or being called by) a number of procedures which, while related, are not structurally isomorphic. The use of descriptive process linkage is closely related to the notion of "loosely coupled computation" [Winograd, semipublic communication]. The exact procedure and calling convention employed for some particular descriptive linkage may thus be determined *contextually*, based on the structural properties of those process tokens which are currently descriptively accessible. This mechanism also facilitates a "graceful degradation" of the specificity of descriptive procedural linkages, allowing the execution of highly appropriate specialized tokens whenever such tokens are descriptively accessible while still permitting the activation and execution of more generally applicable processes when no specialized token is descriptively accessible. The access characteristics of active memory thus permit the employment of *flexible* calling and binding criteria suggestive of the use of "preference semantics" [Wilks, 1973]) in natural language comprehension.

It should also be noted that the independence of the activation and execution processes, along with the association of process tokens embodying descriptions at more than one level of generality with all executable code, allows the use of executable tokens and their descriptively accessible relatives for such "meta-procedural" functions as problem decomposition, the planning of cognitive strategies, and reasoning about what is known. Internal processes, like objects, can be represented at various abstractive levels, allowing the utilization of schemata which describe different procedural constructs at several levels of generality. Specific execution sequences, for example, may be relaxed in a schematic description to describe *partial* constraints on the execution order of various process modules. As indicated earlier, the task of specializing and elaborating procedural tokens to form integrated routines applicable to particular situations itself involves problems which may be interpreted within the conceptual matching framework. The mechanisms described in the following subsection are thus applicable not only to the external *application* task, but to the internal subproblems of a conceptual matching system as well.

3.2 Mechanisms for Conceptual Integration

As we saw in Section 2, both the description-building and matching components of a conceptual matching task must in general effect the integration of "high-level" guiding conceptual frameworks and "low-level" user-provided details through the discovery of structural and abstractive links. In Subsection 2.3, a sketch of the functional dynamics of conceptual integration was presented which involved the interaction of a "top-down" process for specializing and elaborating guiding frameworks, and a "bottom-up" process for generalizing and interconnecting observed details. This section will illustrate certain important mechanisms involved in the proposed implementation of these two processes within our proposed system. To exemplify these mechanisms, we will outline of a typical sequence of steps by which an integrated description of a matchable entity might be constructed, using several sample schemata from the job placement domain, outlined in Figure 3.2. As indicated earlier, similar processes would be involved in a typical pairing task.

In the context of the job placement task, the Worker schema which is (partially) sketched in Figure 3.2, describing certain expected elements associated with professions of any sort, might typically be available to guide the description of the occupation of a job-seeking user. The matching session might thus begin with the activation of a Job-Seeker-T token having Worker as its sole template:

```
Job-Seeker-T
    (a Worker)
```

```
Worker
      <SELF (a Person)>
      <Equipment (a Machine)>
      <Output { (a Good-or-Service)>
                 (the Product from
                      (the Current-Employer from Worker)
                      viewed-as Manufacturer) } >
      <Current-Employer (a Company)>

Professional-Programmer
      <SELF { (a Person)
              (a Worker) }
      <Principal-Language (a Programming-Language)>
      <Last-System-Implemented
           (a System with
                 Implementor = (the SELF from Professional-Programmer) ) >

System
      <SELF (a Thing)>
      <Implementor (a Person)>
      <Data-Objects (a Program-Construct)>
      <Application (an Application)>
      <User (a Possibly-Fictitious-Person)>

Business-System
      <SELF { (a Thing)
              (a System with Application = (a Business-Application) ) } >

Internal-Business-System
      <SELF { (a System with
                   User = (the Employer
                                from (the Implementor
                                     from Internal-Business-System
                                     viewed-as System)
                                viewed-as Worker) )
              (a Business-System) } >

Computer
      <SELF { (a Thing)
              (a Machine) } >

Program
      <SELF { (a Thing)
              (the Output
                   from (a Professional-Programmer)
                   viewed-as Worker) } >

COBOL
      <SELF (a Programming-Language)>

Inventory-Control-Application
      <SELF (an Application)>
      <Inventory-Items { (a Things)
                         (a Widgets) } >
```

Figure 3.2
Selected Example Schemata

```
Manufacturer
      <SELF (a Company)>
      <Product (a Good-or-Service)>

J-C-Whitney
      <SELF { (a Company)
               (a Manufacturer with
                    Product = (an Auto-Parts) ) } >

Auto-Parts
      <SELF { (a Thing)
               (a Widgets) } >
```

Figure 3.2, continued
Selected Example Schemata

(By convention, tokens will be distinguished with a "-T" suffix appended to their names.) Upon initiating the execution of an appropriate domain-specific supervisor, a sequence of initial queries would attempt to fill the slots of the template of this token. Let us assume that the system first tries to determine the principal tool used by the job-seeker in his profession, to which the user answers "computer" (perhaps as a single-word response), allowing the elaboration of Job-Seeker-T by constraining the Equipment slot from its Worker template. In its next query, the system might inquire about the characteristic *output* associated with job-seeker's occupation, receiving the word "program" in reply and elaborating the Output slot accordingly. In a similar manner, the system might find that the job-seeker is currently employed by J. C. Whitney, Inc., filling another slot of the Worker schema. Additionally, though, the domain-specific supervisor might judge it useful to make any information associated with the employer more immediately accessible to descriptive access (by various subsidiary processes) through the activation of a token representing a conceptual instance of the firm. After making explicit several characteristics of this newly activated token, active memory would have the following form:

```
Job-Seeker-T
     (a Worker with
          Equipment = (a Computer)
          Output = ( a Program)
          Current-Employer = { J-C-Whitney-T
                                 (a J-C-Whitney) }
J-C-Whitney-T
     (a J-C-Whitney-T)
     (a Manufacturer with
          Product = (An Auto-Parts))
     (the Current-Employer from Job-Seekar-T
          viewed-as Worker)
```

It should be noted that our illustrations of the contents of active memory include only a sample of those tokens which would typically be activated. In particular, we have not attempted to represent the *executable* tokens which would in fact be involved in the description-building sequence which we are outlining. It is also worth noting the bi-directional linkage between the two tokens which are currently active in the above illustration, although such two-way connections will not always be observed.

The particular constraints now imposed on the Equipment and Output slots suggest a specialization of the general Worker framework: the Professional-Programmer. Taken separately, these elaborating details might not constitute strong evidence that the job-seeker was in fact a programmer. A librarian might use the same tool, for example, but to produce an information service, and not a program. A data processing manager, on the other hand, might be said to produce programs as output, but without actually using a computer. In our example, though, the *co-occurrence* of these slot fillers leads to the evocation of a more specific high-level framework. The system thus adds a second, more specialized desriptor to the Job-Seeker-T token. This Professional-Programmer descriptor is now elaborated on the basis of responses to questions suggested by its emergent elements. The Principal-Language slot is filled in much the same manner as the more general Worker slots. In the case of Last-System-Implemented, additional structural detail is added -- a direct reference to Auto-Part-Objects-T, a newly constructed token describing certain constituent elements of the system. Further, our example assumes that the supervisor has chosen to activate and elaborate this token, yielding the following active memory configuration:

```
Job-Seeker-T
     (a Worker with
          Equipment = (a Computer)
          Output = (a Program)
          Current-Employer =
               { J-C-Whitney-T
                 (a J-C-Whitney) } )
     (a Professional-Programmer with
                Principal-Language = (a COBOL)
                Last-System-Implemented =
                      (a System with
                            Implementor = Job-Seeker-T
                            Data-Objects = Auto-Part-Objects-T) )
      J-C-Whitney-T
           (a J-C-Whitney)
           (a Manufacturer with
                Product = (An Auto-Parts))
           (the Current-Employer from Job-Seeker-T
                viewed-as Worker)
      Auto-Part-Objects-T
           (an Auto-Parts)
           (a widgets)
           (the Data-Objects
                from (the Last-System-Implemented from
                            Job-Seeker-T
                            viewed-as Professional-Programmer)
                viewed-as System)
```

Although the J-C-Whitney-T and Auto-Part-Objects-T tokens are constructed and activated in the course of a directed inquiry sequence, they are not immediately useful in specializing the top-level framework, and are introduced instead to facilitate the "bottom-up" component of the description-building task. It is the descriptively "shallow" accessibility of certain relevant information which now permits our hypothetical system to detect a potentially useful conceptual connection between selected elements associated with the currently active tokens. Specifically, the system notices at this point that the token which was activated upon its elicitation as the Data-Objects of our programmer's last completed system is, independently, descriptively accessible as the product manufactured by the same programmer's employer. This observation leads to the goal of *interconnecting* the Auto-Part-Objects-T and J-C-Whitney-T tokens within a higher-level token which express the their relationship in a form likely to be useful to the conceptual integration task. Such a relationship would be established, for example, if the last completed system were found to be a structural engineering program used to analyze the effects of stresses on the manufacturer's products. This explanation, though, might be judged unlikely by the system on the basis of a previously elicited "detail" -- scientific applications of this sort are not typically the domain of a COBOL programmer. Direct user queries might also rule out as well the possibility of a market analysis program which reported sales of different auto parts in various geographical areas, revealing instead an inventory control application involving the maintenance of appropriate stocks of the various products sold by the firm. The system is thus able to construct and activate the token Inventory-Control-Application-T, which provides a higher-level connection between the program application area and the product manufactured by the firm which employed its implementor (which are each viewed in this context as low-level details). At this point, active memory may be illustrated with the following "snapshot":

```
      Job-Seeker-T
           (a Worker with
                Equipment = (a Computer)
                Output = (a Program)
                Current-Employer =
                      { J-C-Whitney-T
                        (a J-C-Whitney) } )
           (a Professional-Programmer with
                Principal-Language = (a COBOL)
                Last-System-Implemented =
                       (a System with
                             Implementor = Job-seeker-T
                             Data-Objects = Auto-Part-Objects-T)
                             Application =
                                  { (an Application)
                                    Inventory-Control-Application-T } ) )
```

```
Inventory-Control-Application-T
     (an Application with
          User = J-C-Whitney-T )
     (an Inventory-Control-Application with
          Inventory-Items =
               (the Product
                    from J-C-Whitney-T
                    viewed-as Manufacturer))
```

The introduction of this connecting token also initiates an attempt on the part of the system to better characterize the programmer's function in his current firm by describing the business purpose for which the software in question was developed by his employer. To this end, the Inventory-Control-Application-T token is generalized, permitting the addition of a Business-Application descriptor within the Application slot of Last-System-Implemented. A computer system designed for a business application is in turn recognized as a particular kind of system described by the Business-System schema, which is added as a second template of the Last-System-Implemented token. Procedures factored with this more general token are then executed to find the company for which the system was implemented, which is assumed (possibly subject to user confirmation) to be the filler of the user slot associated with the more general descriptor based on the System schema. The subsequent description of this user, J-C-Whitney, as the current employer of the system's implementor, leads to the more specific description of Last-System-Implemented as an Internal-Business-System, a category which we will assume to be particularly salient to the current conceptual matching task. At this point, the Inventory-Control-Application-T token, having already served its purpose, may be deactivated, leaving the following active memory configuration:

```
Job-seeker-T
     (a Worker with
          Equipment = (a Computer)
          Output = (a Program)
          Current-Employer =
               { J-C-Whitney-T
                 (a J-C-Whitney) } )
     (a Professional-Programmer with
          Principal-Language = (a COBOL)
          Last-System-Implemented =
               (a System with
                    Implementor = Job-seeker-T
                    User = { J-C-Whitney-T
                             (the Current-Employer
                                  from Job-Seeker-T
                                  viewed-as Worker) }
                    Data-Objects = Auto-Part-Objects-T)
                    Application =
                         { (an Application)
                           (a Business-Application)
                           Inventory-Control-Application-T } )
               (a Business-System)
               (an Internal-Business-System)
```

While space does not permit us to trace the further elaboration and specialization of this description, it is not difficult to imagine an In-House-Business-Systems-Programmer schema which might be introduced at this point as a more specific template of the top-level Job-Seeker-T token, and whose emergent features could be used in building a very detailed description of further characteristics salient to the matching application.

References

Bobrow, D. G., "Dimensions of Representation", in *Representation and Understanding: Studies in Cognitive Science*, San Francisco, Academic Press, 1975.

Bobrow, D. G., and Norman, D. A., "Some principles of memory schemata", in Bobrow and Collins (eds.), *Representation and Understanding: Studies in Cognitive Science*, San Francisco, Academic Press, 1975.

Bobrow, D. G., and Winograd, T., "An Overview of KRL, a Knowledge Representation Language", Stanford Artificial Intelligence Laboratory Memo AIM-293, 1976.

Minsky, M. L., "A framework for representing knowledge", in Winston (ed.), *The Psychology of Computer Vision*, New York, McGraw-Hill, 1975.

Moore, J., and Newell, A., "How can MERLIN Understand?", in Gregg (ed.), *Knowledge and Cognition*, Baltimore, Lawrence Erlbaum Associates,,1973.

Norman, D. A., and Bobrow, D. G., "On data-limited and resource-limited processes", *Cognitive Psychology*, 1975, 7, 44-64.

Pattee, (ed.) *Hierarchy Theory*, 197<n>

Schank, R. C., "The structure of episodes in memory", in Bobrow and Collins (eds.), *Representation and Understanding: Studies in Cognitive Science*, San Francisco, Academic Press, 1975.

Schank, R. C., and the Yale AI Project, "Sam – A story understander", Yale University Computer Science Research Report #43, August, 1975 (a) Simon, H. A., *Science of the Artificial*, 19<nn>

Shaw, D. E., "A Computational Model Of Mass Media Understanding", *Proceedings of the Conference on Structural Learning*, Philadelphia, 1975.

Shaw, D. E., "A Strategy for Making Computers Understand", *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, San Francisco, 1975(a).

Wilks, Yorick, "Preference Semantics", Stanford Artificial Intelligence Laboratory Memo AIM-206, July 1973.

Winograd, T., "Frames and the declarative-procedural controversy", in Bobrow and Collins (eds.), *Representation and Understanding: Studies in Cognitive Science*, San Francisco, Academic Press, 1975.