University of Pennsylvania
ScholarlyCommons

Technical Reports (CIS) | Department of Computer & Information Science

September 1989

The Relevance of Connectionism to AI: A Representation and Reasoning Perspective

Lokendra Shastri
University of Pennsylvania

Follow this and additional works at: https://repository.upenn.edu/cis_reports

Recommended Citation

Lokendra Shastri, "The Relevance of Connectionism to AI: A Representation and Reasoning Perspective", . September 1989.

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-05.

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_reports/774
For more information, please contact repository@pobox.upenn.edu.

The Relevance of Connectionism to AI: A Representation and Reasoning Perspective

Abstract

It is generally acknowledged that tremendous computational activity underlies some of the most commonplace cognitive behavior. If we view these computations as systematic rule governed operations over symbolic structures (i.e., inferences) we are confronted with the following challenge: Any generalized notion of inference is intractable, yet our ability to perform cognitive tasks such as language understanding in real-time suggests that we are capable of performing a wide range of inferences with extreme efficiency - almost as a matter of *reflex*. One response to the above challenge is that the traditional formulation is simply inappropriate and it is erroneous to view computations underlying cognition as inferences. An alternate response - and the one pursued in this paper - is that the traditional account is basically sound: The notion of symbolic representation *is* fundamental to a computational model of cognition and so is the view that computations in a cognitive system correspond to systematic rule governed operations. However, there is much more to a computational account of cognition than what is captured by these assertions. What is missing is an appreciation of the intimate and *symbiotic* relationship between the nature of representation, the effectiveness of inference, and the computational architecture in which the computations are situated. We argue that the structured connectionist approach offers the appropriate framework for explicating this symbiotic relationship and meeting the challenge of computational effectiveness.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-89-05.

This technical report is available at ScholarlyCommons: https://repository.upenn.edu/cis_reports/774

The Relevance Of Connectionism to AI: A Representation and Reasoning Perspective

MS-CIS-89-05
LINC LAB 140

Lokendra Shastri

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

September 1989

The Relevance of Connectionism to AI: A Representation and Reasoning Perspective

Lokendra Shastri*
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104

Abstract

It is generally acknowledged that tremendous computational activity underlies some of the most commonplace cognitive behavior. If we view these computations as systematic rule governed operations over symbolic structures (i.e., inferences) we are confronted with the following challenge: Any generalized notion of inference is intractable, yet our ability to perform cognitive tasks such as language understanding in real-time suggests that we are capable of performing a wide range of inferences with extreme efficiency - almost as a matter of *reflex*. One response to the above challenge is that the traditional formulation is simply inappropriate and it is erroneous to view computations underlying cognition as inferences. An alternate response - and the one pursued in this paper - is that the traditional account is basically sound: The notion of symbolic representation *is* fundamental to a computational model of cognition and so is the view that computations in a cognitive system correspond to systematic rule governed operations. However, there is much more to a computational account of cognition than what is captured by these assertions. What is missing is an appreciation of the intimate and *symbiotic* relationship between the nature of representation, the effectiveness of inference, and the computational architecture in which the computations are situated. We argue that the structured connectionist approach offers the appropriate framework for explicating this symbiotic relationship and meeting the challenge of computational effectiveness.

*Supported by NSF grants IRI 88-05465, MCS-8219196-CER, MCS-83-05211, DARPA grants N00014-85-K-0018 and N00014-85-K-0807, and ARO grant ARO-DAA29-84-9-0027.

1 Introduction

It is generally acknowledged that tremendous computational activity underlies some of the most commonplace cognitive behavior. For example, language understanding - a task that we perform effortlessly most of the time - requires solving several subtasks such as recognizing phonemes, disambiguating word senses, parsing, resolving anaphoric references, imposing selectional restrictions, recognizing speaker's plans, and performing various predictive and explanatory inferences. These tasks are fairly complex and in turn require the integration of a wide range of knowledge pertaining to phonetics, prosodics, syntax, semantics, pragmatics, discourse structure, and that nebulous variety conveniently referred to as common sense knowledge.

Within a traditional formulation of AI and cognitive science most of the above computations are viewed as *inferences*, i.e., systematic rule governed operations over symbolic structures[1]. However, if one accepts such a view of cognition one is immediately confronted with the following puzzle:

> Any generalized notion of inference is intractable, yet our ability to perform cognitive tasks such as language understanding in real-time suggests that we are capable of performing a wide range of inferences with extreme efficiency.

One response to the above puzzle is that the traditional formulation is simply inappropriate and it is erroneous to view computations underlying cognition as systematic rule governed operations over symbolic structures.

An alternate response - and the one pursued in this paper - is that the traditional account is basically sound: The notion of symbolic representation *is* fundamental to a computational model of cognition and so is the view that computations in a cognitive system correspond to systematic rule governed operations. However, such an account is grossly incomplete: There is much more to a computational account of cognition than what is captured by these assertions. What is missing is an appreciation of the intimate and *symbiotic* relationship between the nature of representation, the effectiveness of inference, and the computational architecture in which the computations are situated. We argue that the structured connectionist approach offers the appropriate framework for explicating this symbiotic relationship and meeting the challenge of computational effectiveness.

[1]This is not to say that inference and systematic rule governed operations are the same - inference presupposes the existence of some semantic justification that dictates which operations constitute correct inference. In the present context however, the *nature* of inference is not an issue and what is being said applies to deduction, probabilistic inference, default reasoning, analogical reasoning, etc.

2 Reflexive inference

To underscore the extreme efficiency with which certain inferences need to be drawn in order to support cognitive behavior in real-time, we label such inferences *reflexive*. In the following section we characterize reflexive inferences in terms of their time complexity but informally, these inferences are performed extremely fast - in the range of a few milliseconds to a few hundred milliseconds, they are computed effortlessly, spontaneously, and without deliberation, and the agent does not even become aware (conscious) of the steps involved in arriving at the conclusion. It is as if these inferences are a *reflex* response of the agent's cognitive apparatus - hence the name, reflexive inference.

Reflexive inferences may be contrasted with *reflective* inferences which are relatively slow, and deliberate. In particular, when performing reflective inferences the agent is aware (conscious) of the reasoning process. The surface complexity of the task is not a good indicator of the type of reasoning. Thus a simple task such as "add 391 and 427" requires reflective inferences[2] whereas a complex task such as understanding spoken language (under ordinary circumstances) only requires reflexive inferences.

It may be suggested that reflexive inferences constitute an unusual sort of reasoning, and hence, need not be accounted for in a first pass at developing a computational model of cognition. However, far from being unusual, reflexive inferences make up an overwhelming fraction of all the inferences carried out by human agents (to wit language understanding, vision, and common sense reasoning) and therefore an account of reflexive inference should play a central role even in a *preliminary* computational model of cognition. At stake here are not efficiency, architecture issues, or implementation detail but rather the very viability of the model.

There is one other point that must be clarified. It may be claimed that 'inferences' characterized as reflexive inference are really not inferences. They are either table look-ups or single step inferences with no chaining (i.e., they either correspond to a simple retrieval or can be derived by a single rule-application without any chaining). As the following example will illustrate, such a view underestimates the richness of reflexive inference[3]. Consider a person reading a variation of the Little Red Riding Hood (LRRH) story in which the wolf intends to eat LRRH in the woods. The reader is at the point in the story where the wolf, who has followed LRRH into the woods, is about to attack her. The next sentence reads: "The wolf heard some wood cutters nearby and so he decided to wait." It seems reasonable to claim that the reader will understand this sentence spontaneously and without deliberate thought. However, a careful analysis of this

[2]On the other hand adding two single digit numbers may correspond to a reflexive step.

[3]This example is based on [Schubert 89].

sentence makes it apparent that even though the reader does not become aware of it, understanding this sentence requires fairly elaborate reasoning. This reasoning may (very) informally be described as follows (the 'rules' are in parentheses): To eat LRRH the wolf will have to approach her (because to eat something you have to be near it), if the wolf approaches LRRH she will scream (because a child is scared by an approaching wild animal), if LRRH screams, the wood cutters will hear her (because a loud noise can be heard at a distance and screaming generates a loud noise), if the wood cutters hear the scream they will know that a child is in danger (because a child's screaming suggests that the child is in danger) the wood cutters will come to the location of the scream (because people want to protect children in danger and in part, this involves determining the source of the danger), when the wood cutters see the wolf they will try to prevent it from attacking LRRH (because people want to protect children) in doing so the wood cutters may hurt the wolf (preventing an animal from attacking a child may involve physical force ...) so the wolf will decide to wait (the wolf does not want to get hurt).

Clearly, the above chain of reasoning does not constitute a 'canned' response and in addition to the retrieval of meanings of lexical items, parsing, and resolution of pronominal reference, something equivalent to it must be taking place during the understanding of the sentence in question.

2.1 Time complexity of reflexive inference

Assuming that the number of 'rules' and 'facts' required to encode all relevant aspects of the domain of common sense will easily run into the millions[4], the extremely tight constraint on the time available to carry out reflexive inference entails that the running time of any inference algorithm for performing reflexive inferences can be no *worse* than *sublinear* in the size of the knowledge base. Such a tight constraint introduces a very strong notion of computational effectiveness. Typically, a polynomial time algorithm is considered to be quite 'tractable' in the context of knowledge representation and reasoning [Levesque 89].[5] However, even a polynomial time algorithm is not good enough for modeling reflexive inference - an order n^2, or even an order n algorithm would take far too much time to be of any relevance.

[4]The choice of terminology is not critical and the reader may replace 'rules' and 'facts' by scripts, schemas, frames, constraints, or whatever that might happen to be the readers' favorite way of describing a chunk of knowledge.

[5]In addition to requiring polynomial time, most formulations of limited inference in artificial intelligence preclude the use of *modus ponens* and chaining thereof [Lakemeyer 87] [Frisch & Allen 82]. We consider such an exclusion to be unwarranted: as evident from the LRRH example discussed above, the use of chaining underlies causal and predictive reasoning, and therefore, must be included in an account of reflexive inference.

3 Towards a computational account of reflexive inference

In this section we outline how one can systematically work towards a computational account of reflexive inference. We conclude that a computational account of reflexive inference is one in which inferences can be drawn in a constant number of passes (preferably one or two) of flow of information in a directed acyclic graph where each node in the graph is a processing element and each arc is a hardwired link. This assertion - though straightforward - has a significant impact on the nature of representations and leads to the identification of important constraints on the conceptual structure.

In the following discussion we will often allude to the following three points. These points may be obvious but are, nevertheless, stated here because they are critical to our argument.

1. The notion of a representation is meaningful only in the context of the operations it is capable of supporting. In the case of knowledge representation, any representation must be accompanied by a specification of the retrieval and inferential operations it is expected to support. Unless this is done, one cannot establish the correctness of the representation - let alone judge its goodness.

2. Any computational model of intelligence must be computationally effective, i.e., it should be capable of performing the specified tasks in a specified time frame (which in the case of building cognitve agents is real-time).

3. In view of 2, the specification referred to in 1 must also specify the limits of acceptable performance for the retrieval and inference operations the representation is intended to support. Thus, a specification of the operations that a representation must support and the time frame within which these operations must to be performed, together constitute the *design constraints* for developing a representation.

3.1 Significance of organization: relating structure and inference

The most important step in tackling the problem of reflexive inference (and tractable inference in general) is to:

> Augment the syntax of the representation language so that the *form* (i.e., the syntactic structure) of the representation directly *mirrors* the *inferential structure* of

the knowledge[6].

A representation that naturally provides the requisite coupling between the syntactic structure of the representation and the inferential structure of domain knowledge is a graph whose nodes correspond to 'units' of information (constants, predicates, concepts, properties, features, frames, or whatever) and whose arcs correspond to inferential dependencies between these units. Adopting such a graphical representation has the interesting consequence that inference reduces to search in a physically instantiated graph. This in itself, of course, does not solve the problem of effectiveness because searching arbitrary graphs is a costly operation. However, once we identify inference with graph search it becomes possible to relate the *effectiveness* of the inference process (search) with the *structural properties* of the representation (graph). For instance, searching a tree or a directed acyclic graph (*DAG*) is cheaper than searching a general graph - especially if the search can be performed in parallel. This suggests that if we desire computational effectiveness our representation should map the domain knowledge into a graph with the following property:

> Portions of the graph that are relevant to the solution of a reflexive inference problem must be trees or *DAG*s.

The *direct* relationship between the structural properties of the representation and the effectiveness of inference reduces the problem of reflexive inference to the problem of choosing appropriate representational primitives; primitives that would impart the required structural properties to the graph encoding the domain knowledge. Within such an approach, answers to questions such as: "Should our epistemological primitives be undifferentiated predicates or should we distinguish between 'concepts', 'relations', and 'attributes' ?" and "Should we use a typed (sorted) logic or not?", depend upon - and follow directly from - a detailed specification of the types of inferences that need to be computed reflexively.

In order to illustrate how the choice representational primitives affects the structural properties of a representation consider two different ways of representing the following knowledge:

- Persons are non-pacifists,
- Republicans and Quakers are persons,
- Quakers are pacifists, and Republicans are non-pacifists.

[6]The above notion is not new and underlies semantic networks, frame languages, scripts, etc. The idea of 'vividness' [Levesque 88] is a special case of this general notion.

First consider a 'class-only' system in which everything is expressed in terms of monadic predicates. Next consider a 'class-property system' which makes a distinction between classes (concepts) and properties. These two representations are illustrated in Fig. 1. Notice how the second representation leads to a *DAG* whereas the first one does not. This is significant because it illustrates that depending on the choice of representational primitives the same underlying knowledge may either lead to an acyclic structure that is amenable to effective inference, or result in a cyclic structure that is not. We are glossing over a number of problems here such as treatment of negation, quantification, necessary versus default properties, to name a few. Our intent here is just to point out that choosing a different set of epistemological primitives (class on the one hand and class and property on the other) can change the structural properties of the representation in ways that significantly impact computational effeciency.

Structural constraints required for supporting reflexive inference may not be as stringent as they might appear. This is because the complete graph need not satisfy these constraints, only subgraphs relevant for solving particular problems need do so. A concrete example of this may be found in the connectionist realization of semantic networks described in [Shastri 88]. There it is shown that property inheritance may be computed extremely efficiently - in time proportional to the depth of the conceptual structure - provided the following requirement is satisfied:

> In order to inherit the value of a property P of a concept C effectively, concepts that lie *above* C and that have information about P attached to them, must form a tree

In general, information about every property is not attached to every concept and hence, the above constraint may be satisfied for a large class of inheritance queries even though the complete *IS-A* hierarchy may not be a tree. As an example, consider the IS-A hierarchy shown in Fig. 2. Assume that the property has-belief - with values pacifist and non-pacifist - applies to the concepts shown in the Fig. 2 and that information pertaining to this property is attached to concepts enclosed in a dark box. Even though the concepts form a tangled *IS-A* hierarchy, the inheritance question: "Is Dick a pacifist or a non-pacifist?" can be answered efficiently because the relevant portion of the graph - consisting of all concepts that lie above DICK and that have information about the property has-belief attached to them - form a tree (see Fig. 3).

Cyclic inferential dependencies can also be reduced by using an extremely fine-grained decomposition of terms so as to reduce the density[7] of inferential dependencies in the knowledge

[7]I.e., the ratio of the number of dependencies to the number of terms in the knowledge base. Here 'term' is used in a general sense and includes - predicates, concepts, properties, features, microfeatures, etc.

base. This suggests that the number of nodes in our graphs will, in general, be extremely large and may run into the millions for any non-trivial domain.

3.2 The Role of Architecture

Once we identify inference with graph search the significance of architecture also becomes obvious. Consider searching a *DAG* with n nodes. A serial search algorithm will take $O(n)$ steps to complete the search. However, if we assume that each node is an active processor that can communicate with all its neighbors then the graph search will only take time equal to the diameter of the graph. Thus the parallel search will be sublinear in the size of the graph. In fact, it will be quite reasonable to assume that the diameter of the graph will only be $O(logn)$.

Based on the above observation we assert that a computational account of reflexive inference would involve

- mapping domain knowledge onto a graph whose nodes correspond to 'units' of information and arcs to inferential dependencies between these units.
- mapping the graph onto a parallel machine by assigning a processor to each node and creating a hardwired link for each arc.
- making appropriate epistemological and ontological choices so that the projection of the graph with respect to a reflexive query (i.e., the subgraph relevant to solving the query) is always a *DAG* so that the answer to the query can be computed in a sweep of information flow through the parallel machine.

3.3 Other desirable architectural properties

In Section 3.1 we observed that in order to support efficient inference the relevant parts of the graph must be acyclic, and to minimize acyclic dependencies the 'units' of information must be extremely fine grained. As a result, the number of nodes in the graph can easily run into the millions. The parallel encoding proposed in Section 3.2 requires that a processor be assigned to each node in the graph. Consequently, the number of processors in the system will also run into the millions giving rise to a massively parallel system.

Besides massive parallelism, what other desirable features should such a computer have? First, in order to fully exploit its massive parallelism, the system must operate without a central controller. Other features that will help in an optimal use of parallelism may be identified by recognizing that the computing resources of any parallel system are used in two ways: i) for task

related information processing and ii) communication. The first of these involves computations directly relevant to the task at hand, while the second - the use of resources for communication - constitutes an overhead that does not contribute *directly* to the task. Clearly, minimizing communication costs would help in maximizing the use of parallelism.

Communication costs have two components: encoding/decoding costs and routing costs. The sender of a message must encode information in a form that is acceptable to the receiver who in turn must decode the message in order to extract the relevant information. This constitutes encoding/decoding costs. Sending a message also involves decoding the receiver's address and establishing a path between the sender and the receiver. This constitutes routing costs.

One may minimize routing costs by positing *fixed* connections between processors and stipulating that any message sent by a processor will always be transmitted along *all* - and only all - links emanating from the processor. This would reduce routing costs to zero because sending a message would require neither the decoding of an address nor the setting up of a path. Stipulating fixed connections, however, would require that processing elements have a high degree of connectivity.

But how can the decoding/encoding costs be minimized? A trivial way of reducing these costs to zero would be to stipulate that messages shall not have any content - if there is no content, there will be nothing to encode or decode and therefore the associated cost will be zero. Such a suggestion may sound frivolous but one can come very close to reducing the encoding/decoding costs to zero by restricting all messages to be *scalars*, i.e., by requiring that a message not have any internal structure, its only information content being its *magnitude*.

To summarize, an appropriate computational architecture for efficient inference should have the following features:

- massive parallelism - a node for each unit of information
- no central controller
- hard wired links and a high degree of connectivity
- scalar messages with no internal structure, only a magnitude
- each processor need only compute a scalar output based on the magnitudes of input messages and transmit it to all the processors it is connected to.

The features listed above directly correspond to the core features of connectionism.

The most compelling argument put forth in favor of connectionism is that it is neurobiologically plausible. However, we arrived at the core features of connectionism - without appealing

to the architecture of the animal brain - by simply recognizing the characteristics of information processing that underlies intelligence. That nature has produced a computational device with a similar set of features is clearly not an accident. The above list of features does not make any reference to an important feature of connectionist models, namely, learning. The arguments offered above are independent of the question of learning and would hold even if learning were an issue. The point we wish to make is that there exists a strong case for connectionism even if learning is not a major issue.

4 Impact of the core features of connectionism

In designing any massively parallel system that operates without a central controller one has to address the problems of control and convergence. In the connectionist model we have also placed the additional constraint that messages be simple scalars (this was done to reduce encoding and decoding costs). On the face of it, it appears that this makes the problem of representation and reasoning even more difficult. We argue, however, that the constraint on the nature of messages is a blessing disguise! It forces us to face up to the really hard problem in knowledge representation and reasoning that must be addressed if we are to arrive at a computational account of reflexive inference. This problem being the determination of appropriate organizational principles, the choice of right epistemological primitives and the indentification of important ontological distinctions.

In this section we discuss the impact of the core features of connectionism on the nature of representation and reasoning. This impact may broadly be summarized as follows:

- Connectionism offers an extremely efficient metaphor for reasoning where inference is reduced to spreading activation in a parallel network. (We discussed this in Section 3.)
- Restricting messages to only being scalars entails that messages do not have any direct symbolic content; the information content of a message is not in "What is being said?" but rather in "Who is saying it?" [8] This requires that representations be explicit, multi-faceted, and fine grained.
- In a connectionist system there is no *distinct* interpreter that mediates retrieval and reasoning. The connection pattern, the weights on links, and the computational characteristics

[8] A message has a magnitude and therefore it actually encodes "Who is saying it and *how loudly*?". The magnitude does play an important role in encoding constraint strengths and evidential/probabilistic knowledge but is not critical to our discussion here.

of nodes not only represent domain knowledge but also encode the retrieval and inferential processes that operate on this knowledge. This state of affairs *forces* a strong coupling between a representation and the inferences that the representation is expected to support. It also requires that representations be vivid and directly mirror the inferential structure of the domain.

- Working within a massively parallel computational architecture helps in identifying novel classes of limited inference that can be performed with extreme efficiency and aids in discovering constraints that must be placed on the conceptual struture in order to support extreme efficiency.

- The connectionist approach suggests alternate formulations of information processing. Thus instead of viewing a knowledge base system as a theorem prover or a production system, one may view it as a system that performs constraint satisfaction, energy minimization, or evidential and probabilistic reasoning. This encourages the use of alternate reasoning formalisms that have not received due attention within AI but may be appropriate for modeling a range of cognitive functions.

4.1 Scalar messages and the absence of an interpreter

The scalar nature of messages in a connectionist network entails that the messages have no direct symbolic content: the information content of a message is not "What is being said" but rather "Who is saying it?"[9] [10].

The above restriction on the information content of a message does not pose a problem if one is only interested in encoding *associations* between concepts - storing associations simply requires spreading activation, for which scalar messages suffice. This restriction, however, becomes critical if one wishes to represent structured knowledge and perform retrieval and inference on this knowledge. It requires that all distinctions - no matter how subtle - must be represented explicitly. Thus, every relevant facet of a concept and every role that a concept may play has to be distinguished and represented explicitly. [11] It also requires that concepts be

[9] A message has a magnitude and therefore it actually encodes "Who is saying it and *how loudly*?". The magnitude does play an important role in encoding constraint strengths and evidential/probabilistic knowledge but is not critical to our discussion here.

[10] The statement "messages do not have symbolic content" should be distinguished from the statement "connectionist systems do not have symbolic content". In our view, connectionist systems do have symbolic content, only the messages don't.

[11] This corresponds to the idea of "exploded cases" discussed in [Cottrell 85].

represented at multiple levels of granularity (resolution) including extremely fine grained levels as well as coarse grained ones. For example, the concept "chair" must be represented at various levels of granularity including a level of detail that would be appropriate for inclusion in the description of a living room ("a chair"), a finer level of detail that would be appropriate if we were describing the chair itself ("an antique leather upholstered ..."), and an even finer level where the details about the arm of the chair would become relevant, and so on. The concept "red" cannot be represented as a unitary concept and it must be represented using exploded values such as "apple red", "rose red", "brick red" etc. Similarly, a role such as "patient" must be represented in an exploded manner by positing distinct roles such as "love-patient", "hit-patient", "give-patient", etc. In traditional knowledge based systems, such distinctions do not have to be *represented* explicitly as this burden may be shifted to the interpreter which may treat the role "patient" differently depending upon the action the role is associated with.

When knowledge is encoded in a connectionist network the usual distinction between the representation (knowledge base) and the processes that operate on it (the inference engine) becomes blurred: the connection pattern, the weights on links, and the computational characteristics of nodes not only represent domain knowledge but also encode the retrieval and inferential processes that manipulate this knowledge. In a connectionist system there is no *distinct* interpreter that mediates the retrieval or reasoning process - the interactions among the nodes directly cause changes in the states of nodes that produce the appropriate results.

The absence of an interpreter and the restriction on the nature of messages *forces* a strong coupling between the nature of representation, the nature of inference that the representation is expected to support, and the degree of efficiency with which these inferences have to be carried out. This is not always the case in traditional approaches to knowledge representation where there is a tendency to isolate control issues (also referred to as performance/implementation issues) from issues of representation and expressiveness.

The coupling of issues related to content, organization, and inference, that results from adopting the connectionist framework, may seem misguided and at odds with conventional wisdom. Why should one conflate issues that have been decoupled - especially when the decoupling leads to a neat division of problems. It is indeed true that decoupling of issues serves an extremely important *pedagogical* and *analytical* purpose. Our goal however, is not simply to understand such systems, but to produce a detailed computational account of such systems. And in order to achieve this goal it may be essential to blur these distinctions and adopt an unified approach. Any representation is there for a purpose and the notion of a representation is meaningless unless accompanied by the specification of the operations it supports and the acceptable complexity of

these operations. Thus understanding which inferences have to made efficiently - i.e., focusing on performance issues - is essential for discovering the principles underlying the organization of knowledge, and for making the correct ontological distinctions and choices.

A coupled approach to choosing representations is commonplace in computer science where it is clearly understood that the development of *efficient* algorithms for solving a problem cannot be decoupled from the development of associated data structures. In fact, the design of an appropriate data structure is often the central step in the process. Once the "correct" data structure is discovered, the algorithm follows by fiat.

4.2 A simple example to illustrate some issues in representation

We now consider a simple example that illustrates how the restriction on the nature of messages and the absence of an interpreter affects the choice of representations.

Let objects in the domain we wish to represent have two intrinsic properties: color (with values red and blue) and shape (with values square and circle), and let *left-of* be a binary relation defined between objects. Let obj-1 and obj-2 be two objects in the domain such that obj-1 is a red square, obj-2 is a blue circle, and obj-1 is to the left of obj-2.

For simplicity we assume that the representation is only expected to support the following simple operations:

- Retrieve an object given its (partial) description. For example, retrieve "obj-1" given "red color" or given "red color and square shape".
- Retrieve the value of a specific property of a specified object. For example, retrieve the color of obj-1 as "red".
- Test whether a given relational tuple is true or false. For example, test whether $left\text{-}of$(obj-1, obj-2) is true or false.

Let us develop a simple connectionist representation for this domain. We begin by assuming that the four property values red, blue, square, and circle are grounded in perception, and hence, are $primitive$ concepts (or microfeatures). Thus we represent each of these property values with a distinct node (Fig. 4a). Each node is a processing element and sends out activation when in an $active$ state. One might suggest that the representation of obj-1 may be taken to be the pattern 1010 - this being the pattern of activation over the set of property values (microfeatures) corresponding to obj-1, and similarly, the representation of obj-2 may be taken to be the pattern

0101. Such a representation, however, is inadequate for a number of reasons that are outlined below.

First, the proposed representation does not really *represent* the objects obj-1 and obj-2; the representation does not distinguish between patterns such as 1010 that correspond to objects explicitly represented in the system, and patterns such as 1001 (a red circle) that correspond to *potential* objects as yet unrepresented in the system. Moreover, the representation does not provide any way of associating a name with a stored object. For example, it is not possible to state that the pattern 1010 corresponds to the object obj-1. These are serious limitations because given the description "red square" - i.e., given the pattern 1010 - the system will be unable to *recognize* this pattern as being that of an *existing* concept - let alone recognize it as being the object obj-1.

Another problem with the current representation is that it makes the representation of composite concepts particularly difficult - if not impossible. The representation does not allow the system to refer to objects except by their *full* descriptions and therefore the representation of a composite object must include the full description of all its component parts. For example, as the system can only refer to obj-1 and obj-2 by their complete descriptions, the representation of *left-of*(obj-1, obj-2) will have to be *left-of*(red square, blue circle). Such a representation would soon become untenable as one tries to describe progressively complex structures - specially, if concepts have many more property values than two. [12]

A simple solution to some of the problems listed above can be obtained by using the basic idea of abstraction. This involves positing a separate node for each object explicitly represented in the system and connecting such a node to all the property values of the corresponding object. Thus we would augment the representation of obj-1 by adding a node 'obj-1' and connecting it to the nodes 'red' and 'square'. The same can be done for obj-2 (Fig. 4b). However, it must be remembered that the node 'obj-1' *does not in itself* represent the red square: obj-1. It represents obj-1 only by virtue of being connected to the nodes 'red' and 'square'; if we disconnect the node 'obj-1' from the nodes 'red' and 'square', it ceases to represent obj-1 (or for that matter anything else).

The above needs to be emphasized because a lack of appreciation of this point lies at the heart of prevalent misconceptions about so called "localist" representations. The presence of a node such as 'obj-1' provides a simple but effective way of distinguishing between the representation of explicitly represented (memorized) concepts such as "the red square: obj-1" and potential

[12] We discussed the above representation in spite of its serious limitations because such "distributed representation" schemes have been proposed and defended as viable representations [Rumelhart & McClelland 86]. See Section 7

objects such as "a red circle". It also makes it possible to recognize and name the description "red square" as "obj-1". All we need to assume is that giving the description "red circle" amounts to activating the nodes 'red' and 'square' which - by virtue of their connections - will activate 'obj-1' but leave 'obj-2' inactive.

Our design, however, is still incomplete. The representation in Fig. 4b does not allow us to distinguish between a "red colored square shaped" object and a "red shaped square colored" object. As far as the representation is concerned both are equivalent. Making this distinction is crucial, particularly, as we expect the representation to handle the task of answering questions such as "What is the color of obj-1?" (this is one of the tasks we set for the representation). In response to this question we would like the system to say "red" (i.e., have the node 'red' active without the node 'square' also becoming active). However, the system cannot do so with the proposed representation. Starting with 'obj-1' there is simply no way of activating 'red' without 'square' also becoming active.

In a traditional semantic network the above problem does not arise because the links in the network can be labeled. In a semantic network, the link between 'obj-1' and 'red' would be labeled "has-color" and the link between 'obj-1' and 'square' would be labeled "has-shape" (refer to Fig. 4b). During retrieval, an interpreter will read these labels and decide which link is appropriate for the given task. If the task is to find the color of obj-1, the interpreter would follow the link labeled "has-color" and arrive at the node 'red'.

In a connectionist network, however, we cannot have labels on links: this would amount to sending messages that encode more than a strength of activation Yet we need a mechanism that would allow messages from 'obj-1' to selectively reach 'red' - and not 'square' - whenever the focus of attention is the color of 'obj-1'. This can be done in a straightforward manner by introducing an extra node that would associate 'obj-1' and 'red' only in the context of the property color, and 'obj-1' and 'square' in the context of the property shape.

A possible solution along the above lines is given in Fig. 5 (c.f. [Shastri 88]). The triangular nodes - called binder nodes - provide the required context. Each binder node associates an object, a property and a property value and becomes active on receiving simultaneous activation from a pair of nodes. To find the color of obj-1, one would activate the nodes 'has-color' and 'obj-1'. The binder node linking 'has-color' and 'obj-1' to 'red' will receive coincident activation along two of its links and become active. As a result, it will transmit activation to 'red' which will then become active. If we need to find an object that is red in color we would activate the nodes 'has-color' and 'red'. This will cause the appropriate binder node to become active and transmit activation to 'obj-1' which will become active, thus completing the retrieval. Finding

a "red square object" will involve activating the nodes 'red' and 'has-color' and 'square' and 'has-shape' which would also cause 'obj-1' to be become active.

Even the simple example being discussed here illustrates how the restriction on the nature of messages and the absence of an interpreter forces certain ontological choices. For instance, it becomes necessary to represent attributes of objects as property value $pairs$ instead of as $features$. Such a decision is not critical in an interpreted system; in such a system the fact "obj-1 is red in color" could have been represented by encoding "red" as a feature of "obj-1" and the information that "red is a color" could have been recorded separately. These two "isolated" pieces of information could have put together by an interpreter during processing to ascertain that the object is red in color.

The representation of relations can be derived by applying techniques - similar to those outlined above. Fig. 6 illustrates how $left\text{-}of$(obj1-obj2) may be represented. We posit a generic node labeled 'left-of' connected to two role nodes - one for each role of the relation(role nodes are depicted as diamond shaped nodes). The actual instance of $left\text{-}of$ is represented using the hexagonal $instancer$ node labeled 'left-of-1'. This instancer node receives inputs from the role nodes of 'left-of' and the corresponding fillers of the roles - in this case 'obj-1' and 'obj-2'. There is a link from 'left-of-1' to 'left-of'. The $instancer$ node performs the function of associating the correct role filler pairs. One can see how activating the two roles of 'left-of' and the nodes 'obj-1'and 'obj-2' will lead to the activation of the node 'left-of-1' and in turn of 'left-of' indicating that $left\text{-}of$(obj-1,obj-2) has been asserted as an instance of the relation $left\text{-}of$.

So far we have only considered the representation of relatively stable (long term) knowledge. It is reasonable to assume that nodes such as 'obj-1' and 'left-of-1' exist (for example, they might be learned over time) in order to represent stable grouping of constituents. However, such a scheme is entirely inadequate if such groupings have to be created dynamically for short durations. The need for establishing such dynamic short-term bindings clearly arises in language understanding, vision, and reasoning. In fact it arises in any situation that involves reasoning with representations that include the use of variables. In connectionist circles this problem is referred to as the *variable binding* problem. [13] Consider the following example involving a simple reasoning step. Assume that a network encodes the rule:

$$\forall(x, y)\,(HIT(x, y) \Rightarrow HURT(y))$$

and facts such as HIT(John,Susan) and HIT(Tom,John) among others. HURT(John) clearly

[13]Some researchers have argued that it may be possible to exhibit interesting cognitive behavior without solving the variable binding problem. For example see [Agre & Chapman 88].

follows from the above knowledge by instantiating the rule with the bindings "Tom" for "x" and "John" for "y" and applying modus ponens. If a connectionist network is to infer HURT(John) it must carry out an *equivalent* computation. In generic terms, it must have a way of activating the representation of HURT() given the activation of the representation of HIT(). Furthermore, it must have a mechanism for establishing bindings for variables 'x' and 'y' in the representation of HIT() and ensuring that the same bindings are induced in the representation of HURT(). The problem gets even more confounded if we wish to chain such inference steps and the bindings have to be propagated faithfully along the chain.

Note that any solution that requires such bindings to be pre-wired is unacceptable: prewiring these bindings would correspond to explicitly representing all possible instantiations of the rule. This is not feasible because the number of instantiations may be too numerous - potentially unbounded. Thus we need the ability to set up these bindings on the fly. As we cannot use links or nodes, it seems natural that we may have to use the temporal dimension to solve this problem, and indeed, we have solved an interesting subclass of this problem using time multiplexing (see Section 6).

The examples discussed above are simple and serve to point out how the restriction that all messages be scalar and the absence of a distinct interpreter, requires that representational and inferential issues be coupled. In the Section 5 and 6 we look at two connectionist systems that perform quite complex reasoning tasks over structured information.

4.3 Convergence

Parallelism does not guarantee speed. In order to support extremely efficient inference, the spreading activation process must converge extremely fast. The computation performed by many connectionist systems corresponds to a *relaxation* process wherein activation circulates in a network until finally a stable network state is obtained. It is difficult to place an upper bound on the convergence time of such systems and even in cases where it is possible to do so, it often turns out to be polynomial in the size of the knowledge base [Derthick 88]. As explained in Section 3.1, however, we require our encoding to be such that subparts of a connectionist network that participate in the solution of a reflexive inference problem correspond to *DAG*s. Therefore, it can be guaranteed that the system will converge in a *constant* number of sweeps of spreading activation across the network. Thus the solution would be computed in time proportional to the *diameter* of the network which in most cases will be *logarithmic* in the size of the knowledge

base[14]. An example of such a system is the connectionist semantic network reported in [Shastri 88] and the connectionist system for rule-based reasoning described in [Shastri & Ajjanagadde 89].

5 A connectionist semantic memory

Reasoning that may be characterized as *inheritance* and *recognition* (classification) within a semantic network plays a central role in language understanding, visual recognition, and commonsense reasoning. Inheritance and recognition are also significant because humans can perform these inferences effortlessly and extremely fast - to wit language understanding in real-time. A connectionist semantic memory that can solve the inheritance and recognition problems with the desired degree of efficiency has been proposed in [Shastri 88]. This work prescribes a mapping from a formal specification at the knowledge level to a connectionist network that can solve an interesting class of inheritance as well as recognition problems in time proportional to the *depth* of the conceptual hierarchy. As the response time is only proportional to the depth of the hierarchy, the system scales gracefully and can deal with large knowledge bases.

In addition to achieving efficient performance, adopting a connectionist approach to the design of a semantic memory leads to two other advantages.

- Attempts at formalizing inheritance and recognition in semantic networks have been confounded by the presence of conflicting property-values among related concepts which gives rise to the problems of exceptions and multiple inheritance during inheritance, and partial matching during recognition. Several formalizations of inheritance hierarchies have been proposed but none of them offer a uniform treatment of multiple inheritance as well as partial/best matching based recognition. The connectionist approach suggested an evidential formalization of conceptual knowledge that lead to a principled treatment of exceptions, multiple inheritance, and recognition based on best/partial match.

- The work resulted in the identification of constraints on the conceptual structure that lead to efficient solutions. One of these constraints was mentioned earlier in Section 3.1. (The constraint specified that in order to inherit the value of some property P of a concept C effectively, concepts that lie *above* C and that have information about P attached to

[14]For example, in the context of inheritance and recognition in a semantic network, the diameter corresponds to the number of levels in the conceptual hierarchy, and is logarithmic in the number of concepts in the knowledge base.

them, must form a tree.) Another constraint imposes a uniformity requirement on property value attachments in the conceptual structure and suggests that the conceptual hierarchy must comprise of several alternate "views" of the underlying concepts if information about property values is to be used efficiently during recognition.

A detailed description of the system may be found in [Shastri 88], a brief specification of the representation langauge is given below.

5.1 An evidential representation language

The knowledge in the semantic memory is expressed in terms of a partially ordered set of concepts (i.e., a *IS-A* hierarchy of concepts) together with a partial specification of the property values of these concepts. The set of concepts is referred to as $CSET$, the partial ordering as $\ll$, and the information about property values of a concept is specified using the distribution function δ, where, $\delta(C, P)$ specifies how instances of C are distributed with respect to the values of property P. For example, $\delta(APPLE, has\text{-}color)$ may be $\{RED = 60, GREEN = 55, YELLOW = 23...\}$. Note that δ is only a partial mapping; an agent may not know δ for many concept property pairs. In general, for a given C and P, an agent may know $\delta(C, P)$ only if this information may prove useful in making inferences about C. In terms of a traditional representation language, knowing $\delta(C, P)$ amounts to knowing - explicitly - the values of P associated with C. For convenience we also make use of the # notation. Thus, $\#C[P, V]$ equals the number of instances of C that are observed by the agent to have the value V for property P and $\#C[P_1, V_1][P_2, V_2]...[P_n, V_n]$ = the number of instances of C observed to have the value V_1 for property P_1, ... and value V_n for property P_n.

In terms of the above notation, the inheritance and recognition problem may be stated as follows:

Inheritance: Given a concept C, a property P, and a set of property values $X = \{V_1, V_2, ...V_n\}$, find a V_i that is the most likely value of property P for concept C. In other words, find V_i such that the most likely value of $\#C[P, V_i]$ equals or exceeds the most likely value of $\#C[P, V_j]$ for any other V_j in X. The inheritance problem: $C = BIRD$, $P = mode\text{-}of\text{-}transportation$, and $X = \{FLY, SWIM, WALK\}$, may be paraphrased as: Is the mode of transportation of a bird most likely to be flying, swimming, or walking?

Recognition: Given a set of concepts, $Z = \{C_1, C_2, ...C_n\}$, and a description consisting of a set of property value pairs, i.e., $DISCR = \{[P_1, V_1], [P_2, V_2], ...[P_m, V_m]\}$, find a C_i such that relative to the concepts specified in Z, C_i is the most likely concept described by $DISCR$. In other words, find C_i such that the most likely value of $\#C_i[P_1, V_1], [P_2, V_2], ...[P_m, V_m]$

exceeds the most likely value of $\#C_j[P_1, V_1], [P_2, V_2], ...[P_m, V_m]$ for any other C_j. If $Z = \{APPLE, GRAPE\}$, $DISCR = \{[has\text{-}color, RED], [has\text{-}taste, SWEET]\}$ then the recognition problem may be paraphrased as: "Is something red in color and sweet in taste more likely to be an apple or a grape?"

The proposed connectionist system solves such *inheritance* and *recognition* problems in time proportional to the depth of the conceptual hierarchy.

5.1.1 Constraints on the Conceptual Structure

The following terms will be used in stating the constraints:

Relevance: Given a concept C and a property P, a concept B is relevant to C with respect to P, if and only if i) $C \ll B$ (i.e, B lies "above" C in the partial ordering), ii) $\delta(B, P)$ is known (i.e., distribution of instances of B with respect to values of property P is known) and iii) there exists no other concept A between C and B for which $\delta(A, P)$ is known.

Projection: Given a concept C and a property P, $CSET/C, P$, the projection of $CSET$ with respect to C and P, is defined to be the set of all concepts X_i such that $C \ll X_i$ and $\delta(X_i, P)$ is known. (I.e., the projection $CSET/C, P$ is the set of all concepts above C whose distribution with respect to the property values of P is known.)

The first constraint on the conceptual structure restricts the nature of the partial ordering of concepts. This restriction is predicated by the nature of property-value attachment in the conceptual structure.

Constraint-1: The conceptual structure must be such that the ordering induced by $\ll$ on the projection $CSET/C, P$ results in a tree.

The above constraint does not require that all concepts be organized as a tree. It only requires that given a property P, all concepts that have values of property P explicitly associated with them should form a tree.

Before we state the second constraint we describe a particular conceptual structure that allows this constraint to be expressed in a relatively simple manner. The proposed conceptual structure also (trivially) satisfies Constraint-1 mentioned above.

In the proposed scheme (see Fig. 7), concepts are organized in a three tier structure. The top most tier consists of a pure taxonomy that classifies the domain concepts into several distinct ontological categories. Such categories are derived using the principle of predicability, [Keil 79, Sommers 65] which says that different sorts of things have different sorts of properties applicable to them, and one may classify things according to the predicates that apply, or do not apply, to them. Sommers has argued that ontological categories should form a strict taxonomy.

The third or the lowest tier of the conceptual structure consists of instances. The second tier consists of a number of taxonomies called views. The root of a view is a leaf of the ontological tree, and the leaves of a view are instances. Multiple views may be defined with the same leaf of the ontological tree as their root and therefore, there may be multiple views that have the same instance as one of their leaves. Thus instances may have multiple parents, however, each parent must lie in a distinct view. The above organization offers advantages permitted by tangled hierarchies by allowing instances to have multiple parents, but retains certain tree like characteristics that helps in simplifying the interactions between information represented in the conceptual structure and eventually leads to a parallelizable solution.

Constraint-2: If $\delta(C_i, P)$ - i.e., distribution of instances of C_i with respect to property P - is known for some concept C_i in view H, then $\delta(C, P)$ should be known for as many concepts in H as may be necessary to ensure that given any token T_j under H, there exists a concept B_j in H that is relevant to T_j with respect to P.

Constraint-2 requires that if information about some property is stored within a view then such information must be stored at a sufficient number of concepts (sufficient in the sense specified in the constraint). This constraint also suggests that if a property value (or a cluster of property values) can be used to discriminate among a set of instances but is of no special significance as far as members outside this set are considered, then one must define a distinct view over this set of instances so that information about this property value may be used efficiently.

If either of the constraints mentioned above is violated, the system will produce anomalous results. These and other issues are discussed at length in [Shastri 88].

6 A connectionist system for rule-based reasoning

The connectionist semantic network described above is capable of representing long term and stable relationships and bindings. However, it does not address the problem of maintaining and propagating dynamic or short-term bindings. For the sake of clarity if we suppress the evidential aspect of reasoning in the system, the system deals with rules of the form

$$\forall(x)P(x) \Rightarrow Q(x)$$

and

$$\forall(x)P(x) \Rightarrow Q(x, a)$$

Although multiple rules participate in a derivation, it is always the case that all variables are bound to the same individual and thus the system can get by without actually solving the dynamic binding problem.

The need for establishing dynamic and temporary bindings clearly arises in inference that involves variables and multi-place predicates. Consider the following example. Assume that a network encodes the rule: $\forall x, y\ [HIT(x, y) \Rightarrow HURT(y)]$ and facts such as $HIT(John, Mary)$ and $HIT(Tom, John)$ among others. $HURT(John)$ clearly follows from the above knowledge by instantiating the rule with the bindings $John$ for y and applying modus ponens. If the network is to infer $HURT(John)$ it must carry out an *equivalent* computation. In generic terms, it must have a way of activating the representation of $HURT$ given the activation of the representation of HIT. Furthermore, it must have a mechanism for establishing bindings for variables x and y in the representation of HIT and ensuring that the same bindings are induced in the representation of $HURT$. The problem gets even more confounded if we wish to chain such inference steps and the bindings have to be propagated faithfully along the chain.

Any solution that requires that such bindings be pre-wired is unacceptable: prewiring these bindings would correspond to explicitly representing all possible instantiations of the rule (this would also mean that the system is only dealing with propositions not with quantified sentences involving variables!). This is not feasible because the number of instantiations may be too many - potentially unbounded. Thus we need the ability to set up these bindings on the fly. This problem has received considerable attention recently (see [Touretzky & Hinton 88] [Smolensky 87][Dolan & Dyer 88] [Shastri & Ajjanagadde 89]). The latter have suggested a connectionist system that can perform a broad class of deductive inference involving variables and multi-place predicates with extreme efficiency. Specifically, the system can represent knowledge expressed in the form of *rules* and *facts* and determine whether a given *query* follows from the facts and rules encoded in the system. If the argument structure of the rules satisfies certain constraints (these are specified in [Shastri & Ajjanagadde 89]), the system responds to queries in *optimal* time, i.e., the time taken to draw an inference is just proportional to the *length* of the proof. The system maintains and propagates several variable bindings over long chains of inference by using a phased clock.

Rules in the system are assumed to be sentences of the form

$$\forall x_1, ..., x_m[P_1(...) \wedge P_2(...) ... \wedge P_n \Rightarrow \forall y_1, ..., y_k \exists z_1, ...z_l Q(...)]$$

where arguments for P_i's are subsets of $\{x_1, x_2, ...x_m\}$, while the arguments of Q may consist of any number of arguments from among the x_i's and any number of constants besides the universally and existentially quantified arguments introduced in the consequent. Notice that the more commonly occurring rules of the form

$$\forall x_1, ..., x_m[P_1(...) \wedge P_2(...) ... \wedge P_n \Rightarrow Q(...)]$$

- where every variable occurring in a rule is universally quantified with the scope of quantification being the entire implication - are just a special case of the more general form specified above. Facts are assumed to be atomic formulas of the form $P(t_1, t_2 ... t_k)$ where t_i's are either constants or (existentially or universally) quantified variables. A query is an atomic formula whose arguments are either bound to constants or are existentially quantified variables. Some examples of rules, facts, and queries follow:

Rules:

$\forall x, y, z\ give(x, y, z) \rightarrow owns(y, z)$

$\forall x, y\ owns(x, y) \rightarrow can - sell(x, y)$

$\forall x\ omnipresent(x) \rightarrow \forall y, t\ present(x, y, t)$

$\forall x, y\ born(x, y) \rightarrow \exists t\ present(x, y, t)$

$\forall x\ triangle(x) \rightarrow number\text{-}of\text{-}sides(x, 3)$

$\forall x, y\ sibling(x, y) \wedge born\text{-}at\text{-}the\text{-}same\text{-}time(x, y) \rightarrow twins(x, y)$

Facts:

$give(John, Mary, Book1)$; John gave Mary Book1.

$give(x, Susan, Ball2)$; Someone gave Susan Ball2.

$omnipresent(x)$; There exists someone who is omnipresent.

$triangle(A3)$; A3 is a triangle.

Queries:

1. $owns(Mary, Book1)$; Does Mary own Book1?
2. $owns(x, y)$; Does someone own something?
3. $can\text{-}sell(x, Ball2)$; Can someone sell Ball2?
4. $present(x, Northpole, 1/1/89)$; Is someone present at the north pole on 1/1/89?
5. $number\text{-}of\text{-}sides(A3, 4)$; Does A3 have 4 sides?
6. $can\text{-}sell(Mary, Ball2)$; Can Mary sell Ball2?

All queries except 5 and 6 follow from the rules and facts and the system answers yes to these queries.

As discussed in Section 3.1, the efficiency requirement imposed by reflexive reasoning entails that the inferential dependencies in the knowledge base be acyclic. In graphical terms what this means is that if we depict each predicate by a unique node and for every rule

$$P_1(...) \wedge P_2(...) ... \wedge P_n(...) \Rightarrow Q(...)$$

in the knowledge base, if we draw directed arcs from the nodes P_is to the node Q, then the resulting graph must be acyclic. This restriction - although quite strong - does allow us to capture a broad range of common sense reasoning situations. For example, it admits restricted

types of *causal reasoning* - reasoning about actions and events wherein there is no circular causality. It also admits terminological reasoning, i.e., reasoning with definitional knowledge of concepts/terms [Ajjanagadde 90].

7 Distributed v/s localist representations

Most of the discussion in this paper was carried out in the context of a particular brand of connectionism, namely, structured connectionism (aka the localist approach to connectionism). The structured approach is to be contrasted with other brands of connectionism that subscribe to the view that intelligent behavior is an emergent property of a large (unstructured) ensemble of simple processing elements. Distributed representation schemes (as against localist schemes) represent a concept as a pattern of activity over a large number of nodes (microfeatures). In [Feldman et al. 88] it has been pointed out that distributed representations suffer from several problems: "cross-talk, communication, invariance, and the inability to capture structure". We faced several of these problems in the process of developing the representation of a simple domain in Section 4.2 and it is clear that $truely$ distributed representation schemes are inadequate. Some distributed representation schemes such as those proposed in [Rumelhart & McClelland 86] work fine for representing an unstructured set of (unstructured) objects. However, from the point of view of AI, the problem of representing a set of entities is of minimal interest. At the least, we require that a knowledge representation system be capable of representing a set of (structured) entities together with some relations defined over these and also support limited retrieval and inferential operations on this knowledge.

Imagine a distributed representation of a domain consisting of the entities: a and b, and relation: $same$ defined by the tuples (a,a) and (b,b). Compare the representations of a, b, and $same(a,a)$ in such a distributed representation. Clearly, we cannot have the representation of a be a pattern ranging over the $entire$ set of nodes and at the same have the representation of $same(a,a)$ also be a pattern ranging over the entire set of nodes. It turns out that a meaningful representation would require a four way partitioning of nodes: one group (ROLE1) to represent the first role of a relation, another (ROLE2) to represent the second role of a relation, a third (REL) to represent the different relations defined over the set of entities, and a fourth to control the association of patterns between the other three groups of nodes [Hinton 81]. However, there are problems even with the above semi-distributed scheme. Consider the representation of same(a,a). For complete generality, let us assume that same(a,a) is represented by a pattern a' in partition ROLE1, pattern a" in partition ROLE2 (the patterns in the other two partitions

are not important here). How should the patterns a' and a" compare among themselves and with the representation of a in other parts of the system? Should a' and a" be identical? If so how does the system enforce that these two representations of a - occurring in different parts of the network - are identical? If not, how does the rest of the system understand that these two patterns refer to the same entity and how does the system relate these two different patterns to the representations of a in other parts of the network. Finally, is there a canonical representation of a somewhere in the system? Questions such as those raised above need to be answered carefully.

At one level, the only difference between a distributed representation and the localist representation is that localist representations use *abstraction* - a fundamental notion in computer science. Thus for every concept, localist representations posit an $additional$ 'focal' node that is connected to all the microfeatures of the concept. The 'focal' node in itself does not represent the concept, in fact, by itself it does not represent anything. A 'focal' node acquires meaning solely by virtue of its connections to the nodes representing the microfeatures of the associated concept and to the 'focal' nodes of other concepts the associated concept is related to. This difference may seem unimportant but turns out to be critical for representing structured knowledge.

8 Conclusion

In this paper we have argued that one can offer a computational account of reflexive inference while working within the traditional framework by properly understanding the coupling between representation and inference, determining appropriate organizational principles for structuring knowledge, and identifying the features required of an architecture appropriate for realizing the required representation and computations. Connectionism appears to be the appropriate computational framework for achieving this goal. Some tangible progress towards this end has been made and it appears that connectionism will play a key role in the development of a detailed and computationally effective model of reflexive inference.

1 References

[Agre & Chapman 88]Agre, P.E. & D. Chapman. Indexicality and the Binding Problem. Presented at the Symposium "How Can Slow Components Think So Fast?" Stanford, 1988.

[Ajjanagadde 90] Ajjanagade V. Forthcoming Ph.D. dissertation. University of Pennsylvania, 1990.

[Cottrell 85] Cottrell, G.W. A connectionist approach to word-sense disambiguation. Ph.D. Dissertation, Dept. of Computer Science, University of Rochester, 1985.

[Derthick 88]Derthick, M., Mundane reasoning by parallel constraint satisfaction, Ph.D. thesis, CMU-CS-88-182, Carnegie Mellon University, Sept. 1988.

[Dolan & Dyer 88]Dolan, C., and Dyer, M., Parallel retrieval and application of conceptual knowledge, Technical Report TR UCLA-AI-88-3, University of California, Los Angeles, Jan. 1988.

[Fodor & Pylyshyn 88] Fodor J.A. and Pylyshyn Z.W. Connectionism and cognitive architecture: A critical analysis. In *Connections and Symbols* Steven Pinker and Jacques Mehler (eds.) The MIT Press, Cambridge, MA. 1988.

[Frish & Allen 82] Frisch, A.M. and J.F. Allen, Knowledge retrieval as limited inference in D.W. Loveland (Ed.), *Lecture Notes in Computer Science: 6th Conference on Automated Deduction*, Springer-Verlag, New York, 1982.

[Hinton 81] Hinton, G.E. Implementing Semantic Networks in Parallel Hardware. In *Parallel Models of Associative Memory* G.E. Hinton and J.A. Anderson (Eds.). Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[Keil 79] Keil, F.C. *Semantic and conceptual development.* Cambridge, MA: Harvard University Press.

[Levesque 89] Levesque, H. Logic and Complexity of Reasoning. KRR-TR-89-2, Computer Science Department, University of Toronto.

[Lakemeyer 87] Lakemeyer G. Tractable meta-reasoning in propositional logics of belief. In Proc. *IJCAI-87* Milano, Italy, 1987. pp. 402-408.

[Rumelhart & McClelland 86]Rumelhart, D.E. & J.L. McClelland, (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Vol I. Cambridge, MA: Bradford Books/MIT Press.

[Schubert 89] Schubert, L.K. An Episodic Knowledge Representation for Narrative Texts. To appear in Proc. *KR-89*, Toronto, Canada.

[Shastri 88] Shastri, L. *Semantic Networks: An evidential formalization and its connectionist realization.* London: Pitman/Los Altos: Morgan Kaufman. 1988.

[Shastri & Ajjanagadde 89] Shastri, L. & V. Ajjanagadde. "A connectionist system for logical inference with multi-place predicates and variable bindings". Technical Report MS-CS-89-06. Computer and Information Science Department, University of Pennsylvania, Jan. 1989.

[Smolensky 87] Smolensky, P., On variable binding and the representation of symbolic structures in connectionist systems, Technical Report CU-CS-355-87, Department of Computer Science, University of Colorado at Boulder, Feb. 1987.

[Sommers 65] Sommers, F. (1965). Predicability. In *Philosophy in America* (ed.) M. Black. Ithaca, NY: Cornell University Press.

[Touretzky & Hinton 88] Touretzky, D. and Hinton, G., A Distributed Connectionist Production System. *Cognitive Science* 12(3), pp 423-466.

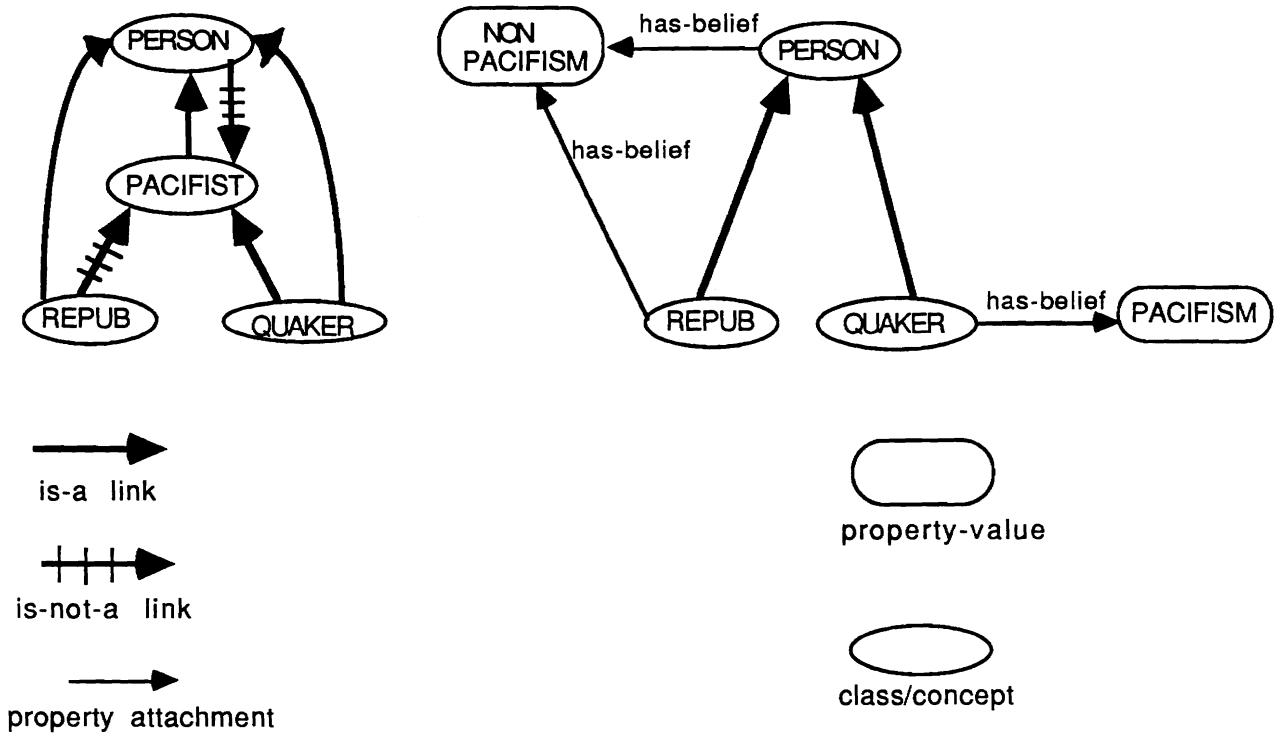


Fig. 1 Two representations: A "class-only" system and a "class-property" system.

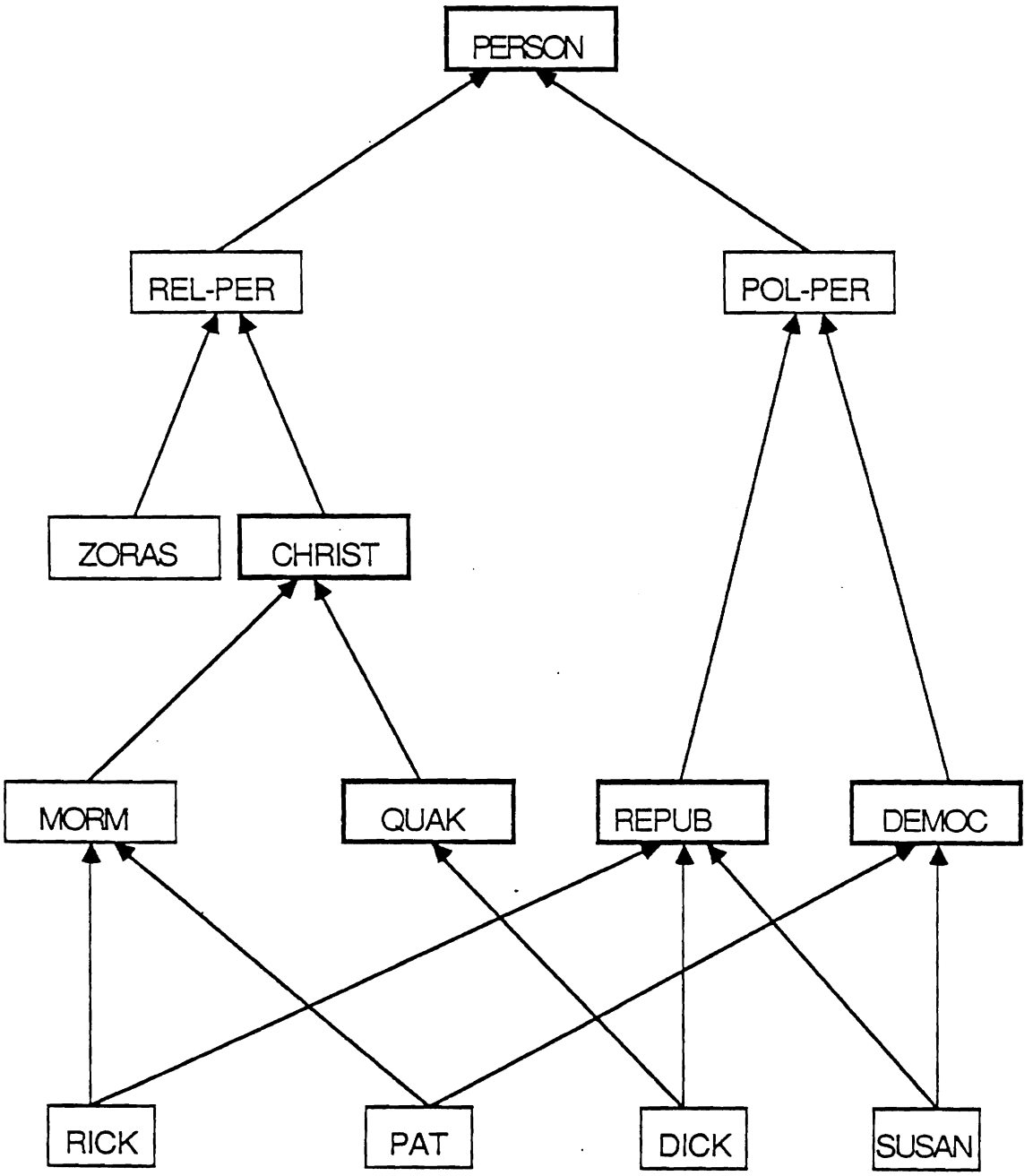


Fig. 2 A tangled conceptual hierarchy

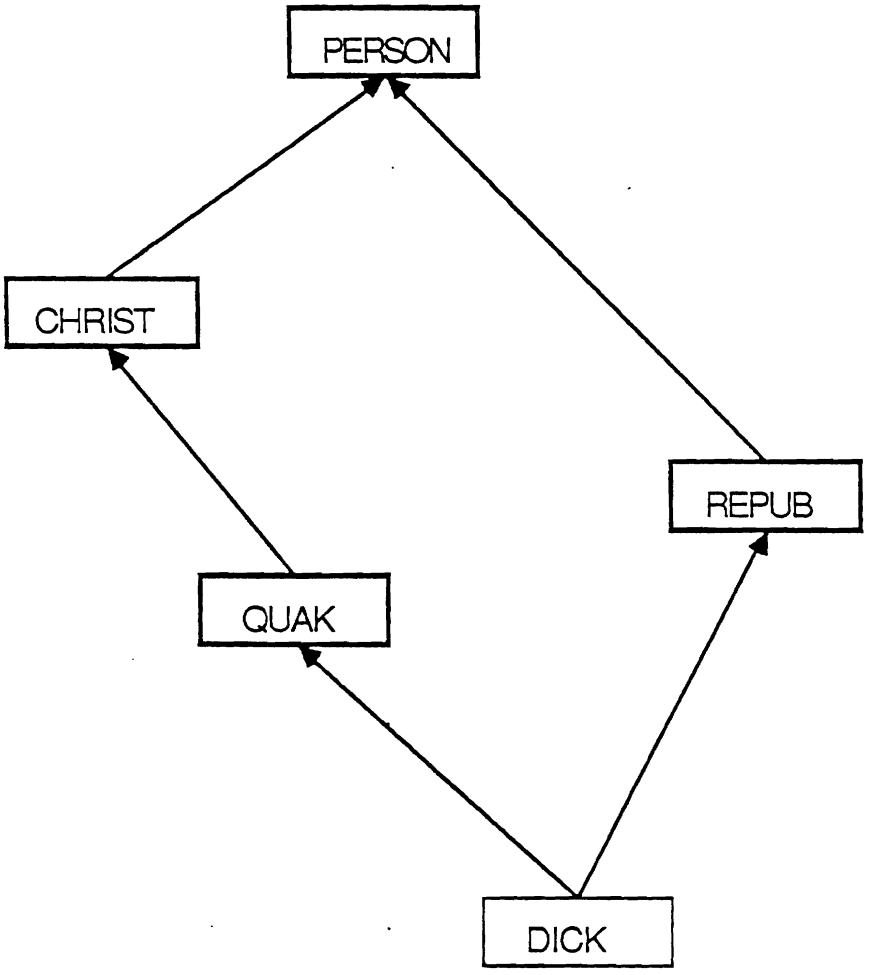


Fig. 3 The subgraph relevant to the inheritance problem: Is Dick a pacifist or a non-pacifist?

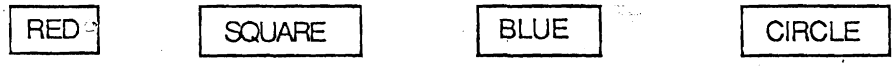
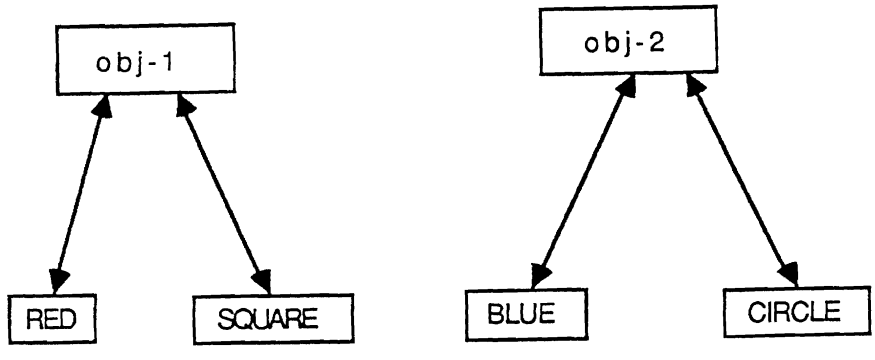


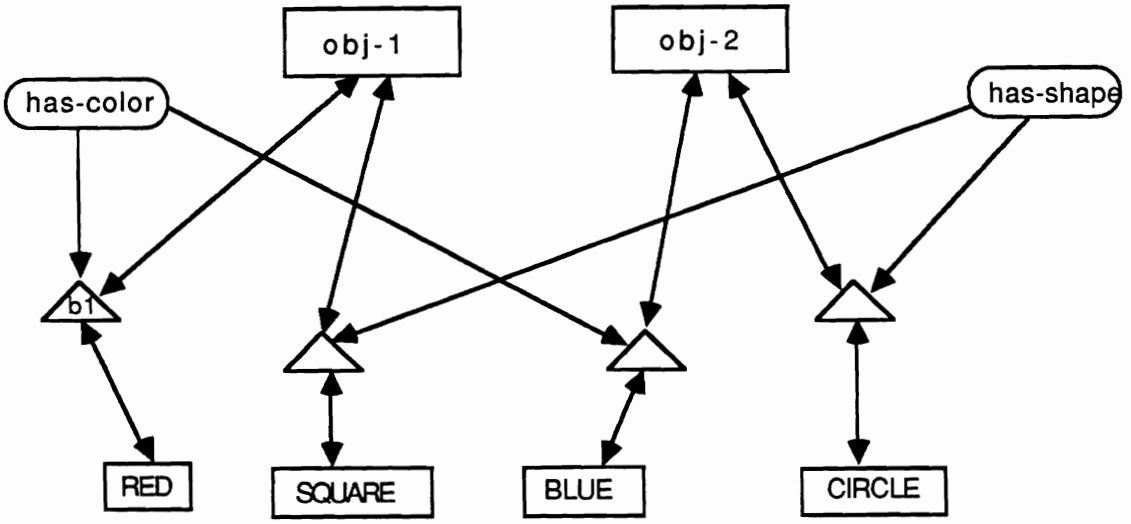
Fig. 4a An unstructured representation of obj-1 and obj-2



obj-1: 'red square'

obj-2: 'blue circle'

Fig. 4b Grouping properties of objects using focal nodes.



Triangular nodes such as $b1$ are binder nodes.

obj-1: 'red square'

obj-2: 'blue circle'

Fig. 5 Representing structured objects using binder nodes.

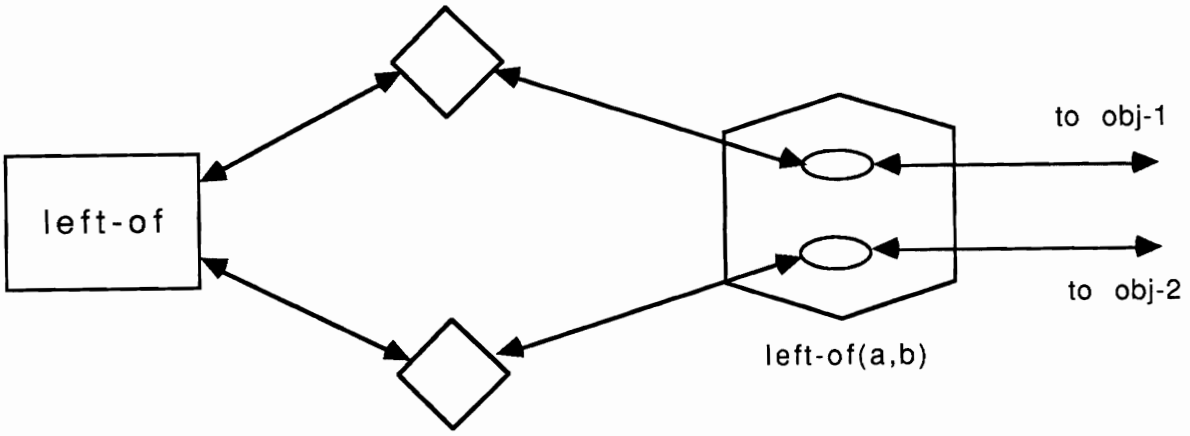
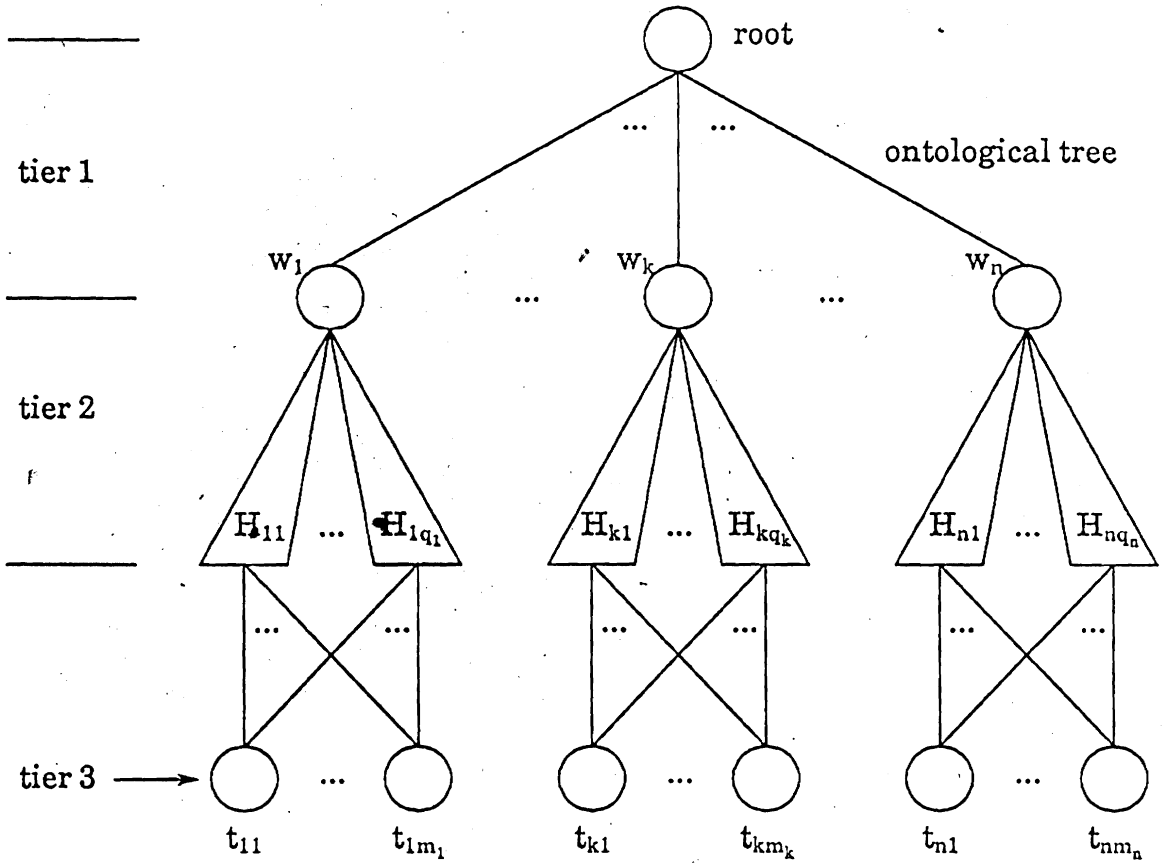


Fig. 6 A representation of left-of(obj-1,obj-2)



t_{11} ..., t_{nm_n} are tokens.
A token may have multiple parents but at most one parent per view.
w_1, ..., w_n are leaves of the ontological tree.
H_{i1}, ..., H_{iq_i} are q_i views defined over tokens of ontological type w_i.

Fig. 7 The multiple views organization