Principles of Knowledge Representation and Reasoning:

Proceedings of the Fifth International Conference (KR '96)

Edited by

Luigia Carlucci Aiello
(Università di Roma
"La Sapienza")

Jon Doyle
(Massachusetts Institute
of Technology)

Stuart C. Shapiro
(State University of New York
at Buffalo)

Morgan Kaufmann Publishers, Inc.
San Francisco, California

Parallel Transitive Reasoning in Mixed Relational Hierarchies

Eunice (Yugyung) Lee and James Geller
Department of Computer and Information Sciences
New Jersey Institute of Technology
Newark, NJ 07102

Abstract

Class hierarchies have been used traditionally in Knowledge Representation and Reasoning for a number of purposes such as inheritance, classification and transitive closure reasoning. In the last several years we have been following two lines of investigation to extend this kind of research. From a conceptual level we have worked on techniques to extend such reasoning to hierarchies other than the IS-A hierarchy. Specifically, we have developed a model of inheritance for part hierarchies (Halper 1992, Halper 1993, Halper 1994). From an implementation point of view we have worked on building fast reasoners based on massively parallel representations of IS-A, Part-of, Contained-in, etc. hierarchies (Lee 1993, Lee 1995, Lee 1996a, Geller 1991a, Geller 1991b, Geller 1993a, Geller 1993b, Geller 1994a, Geller 1994b). However, in all this work we assumed that there exist separate hierarchies. In an often-cited paper by Winston, Chaffin, and Hermann (Winston 1987) a model of reasoning is introduced that permits the combination of IS-A, Part-of, and Contained-in in a single hierarchy. The purpose of our paper is to present representational constructs and reasoning algorithms that combine these three ingredients: mixed relation hierarchies, transitive closure reasoning, and massively parallel algorithms. It is hoped that this combination will lead to progress both in better approximating human commonsense reasoning and in better approximating human speed of reasoning. We conclude the paper with a brief description of a medical vocabulary that we have been using as a source of test data.

1 INTRODUCTION

The use of IS-A hierarchies has been a main staple in Knowledge Representation and Reasoning since Quillian's inception of semantic networks (Quillian 1968). The number of papers that deals with other hierarchies of transitive binary relations is, comparatively, much smaller. However, the Part-of hierarchy has received some attention (Schubert 1979, Schubert 1983, Schubert 1987). Specifically in (Schubert 1987) the importance of transitive closure reasoning in IS-A and Part-of hierarchies has been well explained. We have previously done research on reasoning in Part-of hierarchies, especially inheritance reasoning (Halper 1992, Halper 1993, Halper 1994).

Schubert *et al.*'s approach to transitive closure reasoning has been an inspiration for us for another reason: It permits transitive closure reasoning in constant time, practically independent of the size of the knowledge base. This scalability of reasoning power with growing knowledge bases has been recognized as a very important factor for improving implemented Artificial Intelligence (Hendler 1995). In the past several years we have developed techniques and algorithms for dealing with two weaknesses of Schubert's approach: (1) Update algorithms are not independent of the size of the knowledge base, and (2) Schubert's original approach is limited to trees. We have dealt with problem (1) by using massive parallelism, following the paradigms of Shastri, Kitano, Waltz, Hendler, and Evett (Shastri 1988, Shastri 1989, Shastri 1990, Kitano 1991, Kitano 1991b, Kitano 1991d, Kitano 1991e, Waltz 1990, Evett 1989, Evett 1993a, Evett 1993b). We have dealt with problem (2) by adapting a representation of Agrawal *et al.* (Agrawal 1989, Agrawal 1990) that permits directed acyclic graphs instead of trees. As a result we have developed and implemented massively parallel algorithms for fast retrieval and update in hierarchies of any kind of binary transitive relation (Lee 1993, Lee

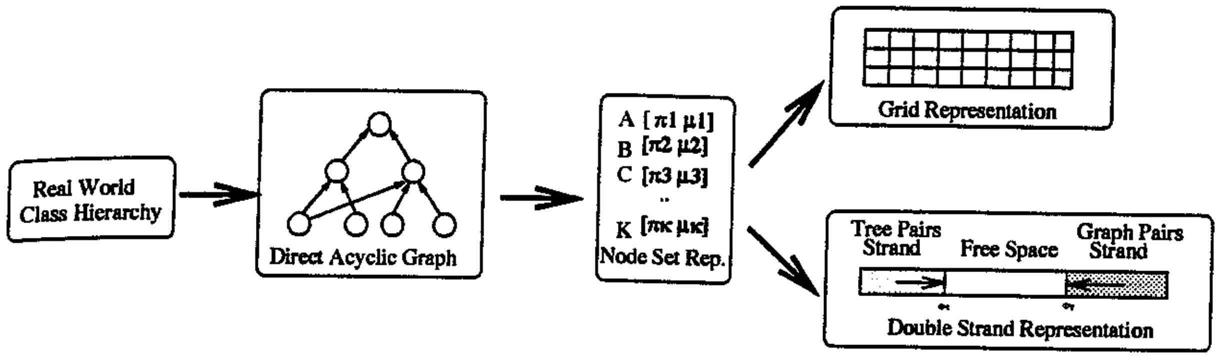


Figure 1: Three Step Mapping

1995, Lee 1996a, Geller 1991a, Geller 1991b, Geller 1993a, Geller 1993b, Geller 1994b, Geller 1994a).

In this paper we are adding another ingredient to our research. In (Winston 1987) it was pointed out that it makes sense to combine different binary transitive relations into a single reasoning process. However, not every conclusion that can be drawn in such a case is correct. Winston *et al.* describe a condition when such reasoning is correct.

As an example, if the following premises are given:

(1) Wings are parts of birds.
(2) Birds are creatures.

We can consider the following two conclusions:

(3) Wings are parts of creature.
(4) Wings are creatures.

We have obtained a reasonable conclusion (3) while (4) is an invalid conclusion (Winston 1987).

Winston *et al.* introduced a hierarchical priority ordering among the hierarchical relations (Winston 1987), such that mixed inclusion relation syllogisms are valid if and only if the conclusion expresses the lower relational priority appearing in the premises. For an alternative approach to transitivity reasoning with mixed relations, see (Cohen 1988).

We have encountered a practical example of this kind of reasoning in our test bed domain which will be described now. J. Cimino *et al.* of Columbia Presbyterian Medical Center have created a large Medical Entities Dictionary (MED) on top of a semantic network model (Cimino 1989, Cimino 1992, Cimino 1993). The MED is a very large vocabulary, incorporating over 43,000 terms, 55,000 IS-A links, 2,500 Part-of links, etc. We have chosen to focus on the InterMED, an offshoot of the MED (Cimino 1994).

The question was posed whether Aspirin can be coated. Aspirin itself would be represented in the MED as a concept. This concept might have several descendants according to different common preparations, such as pills, drops, or capsules. Capsules consist of two parts, the active ingredient and the coating. Thus, the answer is "yes, Aspirin can be coated." In our research, we want to answer a question that a human could answer quickly in a similarly quick manner, avoiding the overhead of a general-purpose reasoner. One way to answer this question quickly within our framework would be to use ***mixed transitivity reasoning*** which combines different hierarchical relations into one single hierarchy. The combined hierarchy would permit a fast positive answer to the given question.

In the next section we will describe our previous work on massively parallel reasoning in detail (Lee 1993, Lee 1995, Lee 1996a, Lee 1996b). In Sections 3 and 4 we will concentrate on the definitions and algorithms necessary for intuitively correct reasoning with mixed relations and for implementing this kind of reasoning within our massively parallel paradigm. Section 5 describes experimental results of our work with the InterMED. Section 6 contains our conclusions.

2 EFFICIENT REPRESENTATIONS FOR CLASS HIERARCHIES

This research has focused on building a massively parallel transitive closure reasoner, called Hydra, that can dynamically assimilate any transitive, binary relation and efficiently answer queries using the transitive closure of all those relations (Lee 1996b). Hydra can dynamically insert new concepts or new links into a knowledge base. Hydra can respond to questions using transitive part relations (Schubert 1987), or to questions of the kind "Is an elephant bigger than a can opener"? if it knows that an elephant is bigger than a person, and a person is bigger than a can opener.

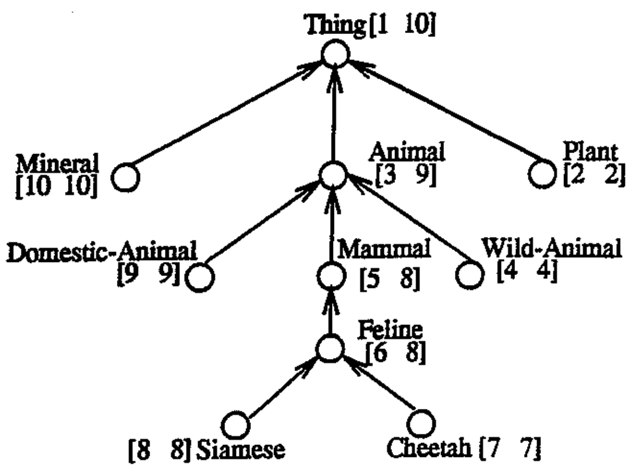


Figure 2: IS-A Hierarchy

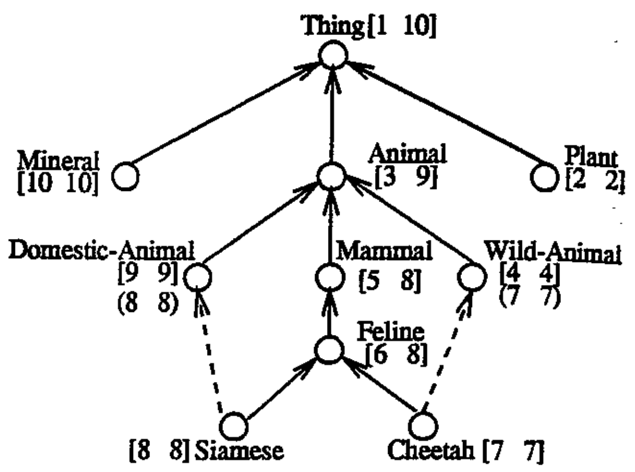


Figure 3: Directed Acyclic Graph of Class Hierarchy

Our tool of choice for achieving fast query and update operations is fine-grained parallelism. This raises the question of how to map the IS-A hierarchy onto the available space of processors. The most obvious intuitive choice is to assign every class of the hierarchy to a single processor. However, this intuitive choice does not carry over to the links between classes (Lee 1996b). We have developed a three step mapping (Figure 1) to deal with this problem.

Step 1: In the first step, an IS-A hierarchy of classes of the real world is mapped onto an isomorphic DAG of nodes, with one class per node. In the simplest possible case this hierarchy is a tree. It consists of nodes, which stand for classes, and connecting arcs, which stand for the IS-A relation. In Figure 2, an example of such an IS-A hierarchy is shown. The IS-A relation is transitive. The node Feline is connected by one IS-A arc to the node Mammal, which means that every Feline is a Mammal. Due to the transitivity of the IS-A relation, every Siamese is also a Mammal, etc.

More interesting than trees are directed acyclic graphs which open the possibility of multiple inheritance. In Figure 3, Siamese has another parent, Domestic Animal. In addition, we have also extended the representation to mixed inheritance hierarchies, i.e., hierarchies that combine relations such as IS-A, Part-of, Contained-in, Greater-than, etc., in one reasoning module.

Step 2: In the second step (Figure 1), the *hierarchy* of nodes is mapped onto a *set* of those nodes, so that every node is annotated with one or more number pairs. This is called the node set representation. The number pair assignment was developed based on Schubert *et al.*'s (Schubert 1983, Schubert 1987) special-purpose reasoner for subclass verification. Schubert *et al.*'s approach (in a tree) permits transitive closure reasoning in constant time, practically independent of the size of the knowledge base. Note that in Figure 2, we may conclude that a Cheetah is an Animal, because [7 7] is contained in [3 9]. Based on techniques that appear in (Schubert 1987, Agrawal 1988), we have extended this work to directed acyclic graphs (DAGs) (Figure 3).

In Figure 3, Siamese is a Domestic Animal because [8 8] is a subinterval (or equal to) (8 8). This is explained in more detail in the next section.

The node set representation makes it easy to represent class hierarchies on arrays of processors. The node set representation is completely order independent, i.e., node sets are used without loss of relevant hierarchy information. This simplifies the parallel update operations necessary to maintain, e.g., a class hierarchy. More details on our node set representation can be found in (Lee 1993, Lee 1996b).

Step 3: In the third step (Figure 1), this node set and the associated number pairs are mapped onto the processor space of a fine-grained parallel computer. We have developed and implemented two methods for mapping this set-based representation onto the processor space of a Connection Machine (initially CM-2, then CM-5). These two representations, the Grid Representation and the Double Strand Representation successively improve transitive closure reasoning in run time and processor space utilization.

In brief, our three step mapping (Figure 1) can be summarized as follows: *class hierarchy* → *directed acyclic*

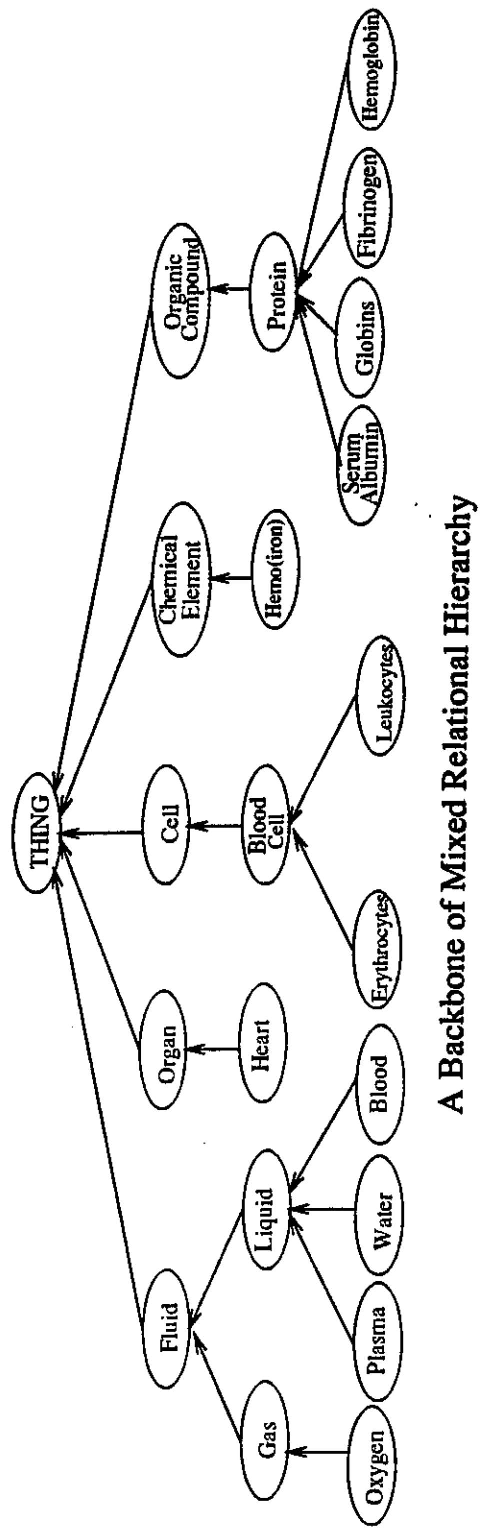


Figure 4: A Backbone of Mixed Relational Hierarchy

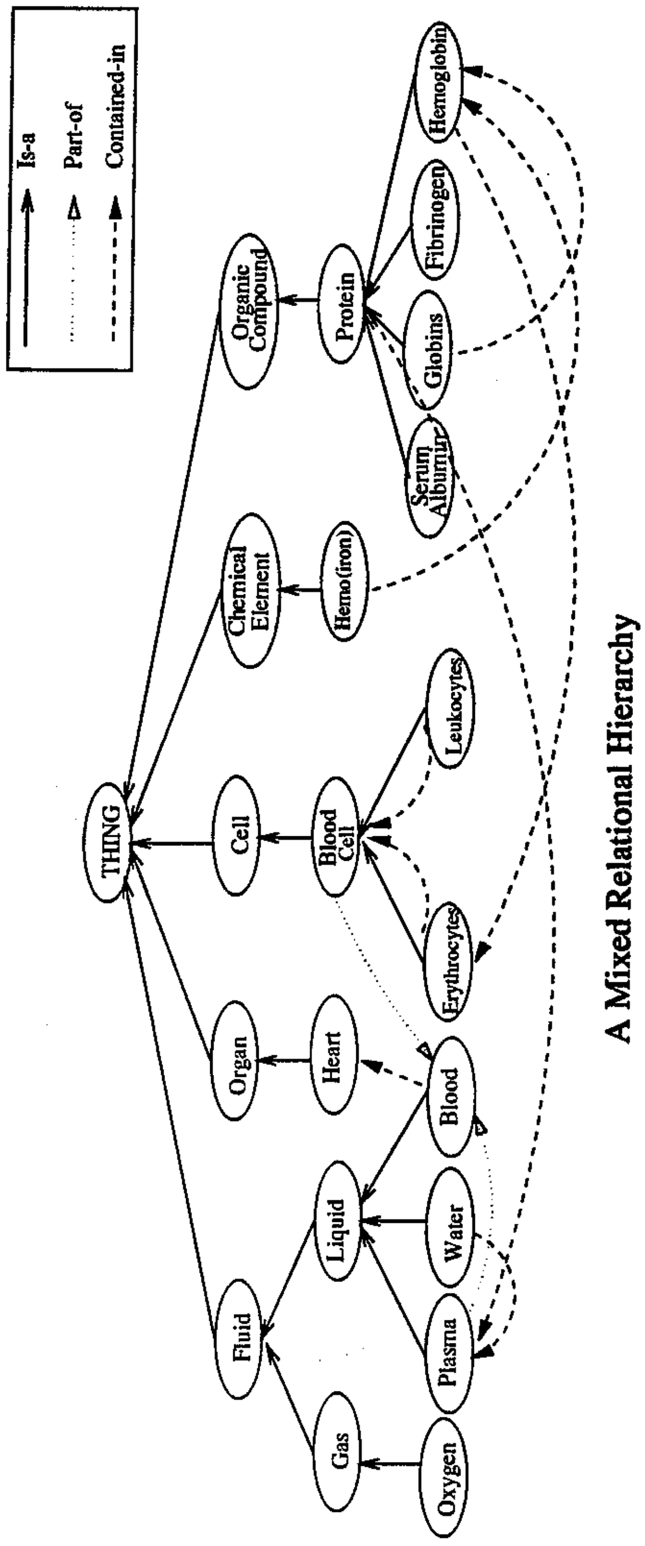


Figure 5: An Example of Mixed Relational Hierarchy

graph → node set + number pairs → processor space. We repeat that as a result of this mapping we get, within certain limitations, constant time responses for transitive closure queries (Lee 1995). In other words, it takes as much time to verify that a Cheetah is an Animal as it takes to verify that a Cheetah is a Feline (Figure 3).

3 EXTENSION TO MIXED RELATIONAL HIERARCHY

Our main idea for constructing a mixed hierarchy representation is to extend our previous approach (Lee 1995) and combine all hierarchical relations into one mixed relational hierarchy. A *mixed relational hierarchy* of the real world is mapped onto an isomorphic directed acyclic graph of nodes. Then a special spanning tree of IS-A relations becomes the backbone of the structure while other hierarchical relations form its branches. Figure 4 shows the backbone and Figure 5 shows the backbone and the branches of an example.

The following questions regarding the mixed relational hierarchy arise: First, how do we distinguish one relation from another? Second, how do we combine them when required? These questions relate not only to the construction of the hierarchy but also the transitivity reasoning. In order to avoid any possible conflict due to a combination of relations, we should design the mixed relational hierarchy to efficiently distinguish one relation from another. Let us consider the example in Figure 6. There, three relations IS-A, Part-of, and Contained-in are involved in a mixed relational hierarchy.

Our representation, extending what was described in Section 2, is generated as follows: (1) Construct an optimal spanning tree of a given DAG such that at every node with multiple parents, we select the link to the parent with the maximum number of weak predecessors. Informally, a weak predecessor Y of a node X is a predecessor which is reachable from X by an upward path containing at least one graph arc and Y is at the head of a graph arc. A graph arc is an arc that was found not to belong to the spanning tree. Assuming a top-down algorithm, all relevant graph arcs are known by the time they are needed. (2) Assign a pair of preorder and maximum number to every node. Preorder numbers are generated by a right-to-left preorder traversal of the spanning tree. The maximum number for every node is the maximum preorder number in the subtree rooted at that node. *Tree pairs* result from this step. (3) All the arcs that are not part of the optimal spanning tree are used to propagate number pairs upward. *Graph pairs* result from this step. (4) The set of all nodes together with the tree pairs and graph pairs is sufficient to represent a class hierarchy. In Figure 6 we use the notation $[\pi \;\; \mu]$ for *tree pairs* and the notation $(\pi \;\; \mu)$ for *graph pairs*. For instance, "Water is a Fluid" exactly because the number pair of Water [8 8] is contained in the pair for Fluid [5 11]. For more details, see (Lee 1993, Lee 1995, Lee 1996a). We need to integrate the different kinds of relations into our numerical representation. For this purpose, we introduce a data element, called "relation type." Each relation is associated with a relation type.

In this paper we assume that the possible relation types are s (for IS-A), p (for Part-of), and c (for Contained-in). As shown in Figure 6, we assign the lowest priority (1) to IS-A, the intermediate priority (2) to Part-of, and the highest priority (3) to Contained-in. In this assignment we follow (Winston 1987). Now we need to define rules how the relation type is assigned to a number pair.

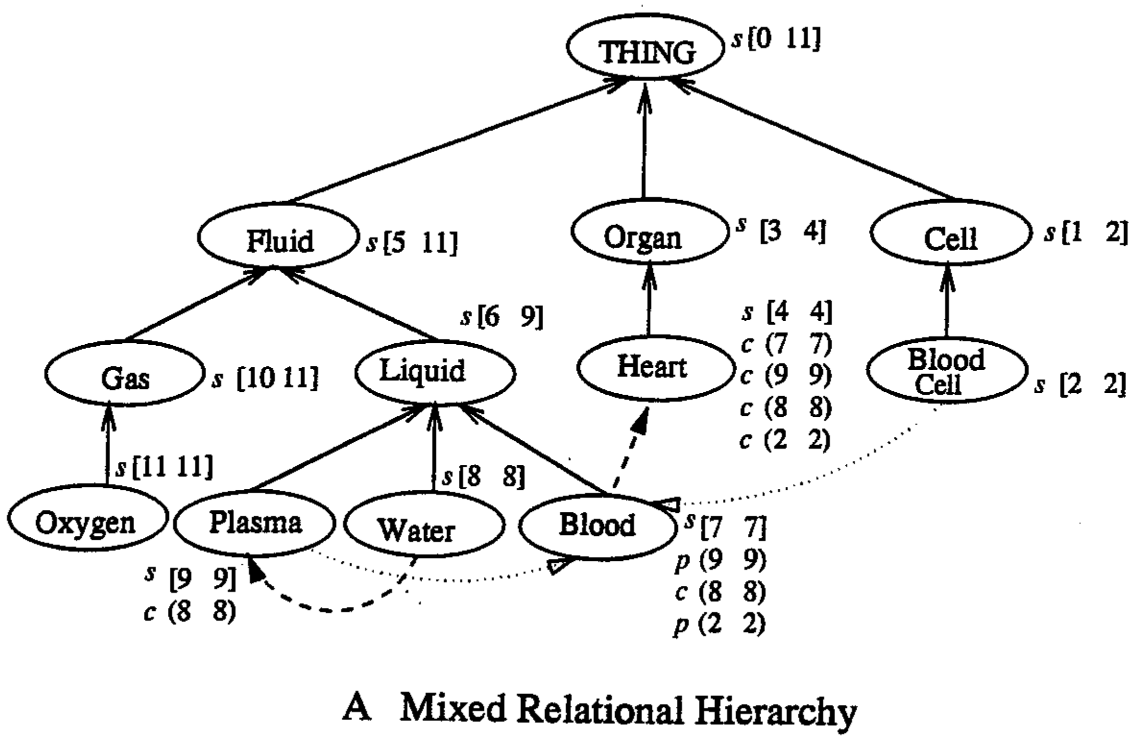
Rule 1: The relation type of a graph pair that was created by propagating a tree pair along one edge is identical to the relation type of the edge.

Rule 2: If a pair with a relation type K with relational priority X is propagated along an edge with a relation type L with relational priority Y then the result

$$R = \begin{cases} K & \text{iff } Y \leq X \\ L & \text{iff } Y > X \end{cases}$$

is the relation type of the pair at the head of the edge.

In our example (Figure 6) assume that a Part-of arc from Plasma to Blood was just inserted. Now the tree pair s[9 9] and the graph pair c(8 8) need to be propagated to the nodes (Blood, Heart) (Lee 1996a). Therefore, this pair s[9 9] is propagated through a Part-of relation from Plasma to Blood, and a Contained-in relation from Blood to Heart. The tree pair s[9 9] of Plasma is propagated through Part-of to Blood, resulting, by Rule 1, in the pair p(9 9). Continuing from Blood to Heart, the pair p(9 9) needs to be changed to c(9 9), by Rule 2. In contrast, the graph pair c(8 8) at Plasma has a Contained-in relation type and its priority is higher than the Part-of relation of the arc from Plasma to Blood and is equal to Contained-in of the arc from Blood to Heart. Therefore, the pair c(8 8) is propagated to Blood and Heart with its own relation type, by Rule 2. At this point, because Heart has a pair c(8 8) that is propagated from Water, we can conclude directly that Heart contains Water, using the



Symbol	Relation	Priority	Relation Type
——→	Is-a	1 (low)	s
⋯⋯▷	Part-of	2 (mid)	p
- - - -▶	Contained-in	3 (high)	c

Figure 6: An Example of Constructing Mixed Relational Hierarchy

relation type c.

4 TRANSITIVE REASONING IN MIXED RELATIONAL HIERARCHY

In this section we show that we can achieve constant time responses (for a given machine size) for parallel transitive closure queries in a mixed hierarchy. Assume that $R_1, R_2, \ldots, R_n$ are hierarchical relations.

Definition 1: A *target of transitivity*, τ, is a node at the end (top) of a path that is used for transitive closure reasoning.

Definition 2: A *source of transitivity*, σ, is a node at the start (bottom) of the path that is used for transitive closure reasoning.

We will now define paths with two different kinds of transitivity.

Definition 3: A path P from σ to τ is *purely transitive* iff $P = \sigma\ R_1\ \sigma_1\ R_2\ \sigma_2\ \ldots\ R_n\ \tau$ and $R_1 = R_2 = \ldots = R_n$.

Definition 4: A path P from σ to τ is *mixed transitive* ($\sigma\ R^x\ \tau$) iff $P = \sigma\ R_1\ \sigma_1\ R_2\ \sigma_2\ \ldots\ R_n\ \tau$ and R^x is such that Priority(x) = Maximum(Priority(R_1), Priority(R_2) , ..., Priority(R_n)).

Both transitivities satisfy the following property: If $\sigma\ R_1\ \sigma_1\ \ldots\ \sigma_n\ R_n\ \tau$ holds, then ($\sigma\ R\ \tau$) & ($R = R_1$ *or* $\ldots\ R = R_n$). Importantly, pure transitivity reasoning and mixed transitivity reasoning can be done in one step. We will present how these mechanisms can be integrated during reasoning.

We now query which relation holds form σ to τ. Our transitive reasoning mechanism works based on an extension of a number pair propagation algorithm introduced in (Lee 1996a). There we proved that if the relation type of the path is IS-A, i.e., there is a tree path from σ to τ, we can achieve the effect of having all graph pairs of σ at τ, without actually propagating these pairs to τ, resulting in an additional saving of space. Above "achieve the effect of having all graph pairs" means that we can perform constant time subclass verification and all operations that rely on subclass verification, including propagation itself. We now have to prove that the algorithm still works after including the concept of relation type.

In Figure 7, the left part shows propagation based on Agrawal's approach (Agrawal 1989) and the right part shows propagation based on our approach. When a graph arc is inserted from Ω to σ with the relation type x, we propagate the pair $x(\pi\ \mu)$ only to σ instead of propagating $x(\pi\ \mu)$ up to all tree predecessors of σ (including τ). Our approach is called Maximally Reduced Propagation and its advantages are explained in (Lee 1996a). This example shows how we reduce the number of graph pairs for all tree predecessors of σ. That the relevant algorithms work was proved in (Lee 1996a). We now show how we can achieve constant time mixed transitivity reasoning for the x relation type from Ω to all tree predecessors of σ (including σ and τ) with the maximally reduced pair propagation.

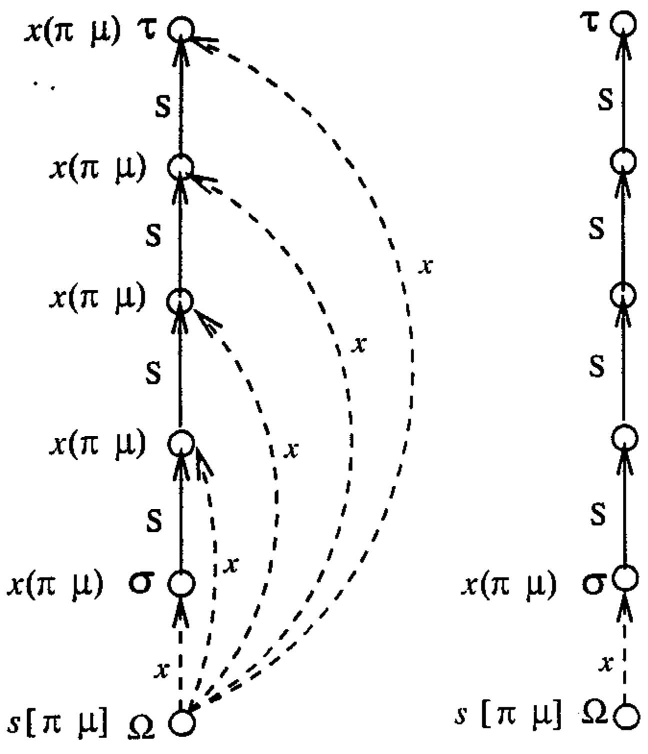


Figure 7: Maximally Reduced Propagation

Lemma 1: Pure Transitivity in IS-A Path
Let R^x be an IS-A relation ($R^x = R^s$). If $\sigma\ R^x\ \tau$ holds through a pure inference path, then after applying our propagation algorithm from (Lee 1996a), the target τ or one of its tree successors will have a number pair with the relation type $x = s$ propagated from σ or from one of its tree predecessors.

Proof: By contradiction, assume that the target τ has a graph pair $s(\pi_s\ \mu_s)$ from the source σ, although an arc A in the inference path from σ to τ is not an IS-A relation. Since the IS-A relation has the lowest relational priority among all relations, the pair $s(\pi_s\ \mu_s)$ can not be propagated through A unless the relation type of $s(\pi_s\ \mu_s)$ changes to the relation type of the arc A (Rule 2). Therefore, the pair $(\pi_s\ \mu_s)$ cannot be associated with the relation type $x = s$. This results in a contradiction. ■

Lemma 2: Mixed Transitivity
Let R^x be a relation. If a source σ relates by R^x to a target τ because of a mixed inference path from σ to τ, the target τ or its tree successor must have a number pair with relation type x propagated from the source σ or its tree predecessor.

Proof: By using Rule 2, this is trivial. ■

Theorem 1: If τ (or a tree successor of τ) has a pair that contains (or is equal to) a pair from σ (or a tree predecessor of σ) then the relation type of the pair of τ (or a tree successor of τ) tells the relation which actually holds between σ and τ.

Proof: We prove it by using the above two lemmas. If the relation is an IS-A relation, the query is an instance of IS-A pure transitivity reasoning (Lemma 1). Otherwise, the query is an instance of mixed transitivity reasoning or pure transitivity reasoning with other relations (Lemma 2). ■

Now we will show how to answer mixed and pure transitivity queries within our paradigm in parallel. We divide pure transitivity into two subcases: one for an IS-A relation and another for other hierarchical relations. A formal description of pure transitivity with other relations will be omitted for space reasons. For more details, see (Lee 1996b). Our massively parallel Double Strand Representation (Section 2) uses pairs of adjacent processors to represent a sequence of graph pairs. In each pair one processor has an odd processor ID and is used to represent a node which propagates its tree pair and its right adjacent processor has an even processor ID and is used to represent a node from which a number pair is propagated (Figure 8).

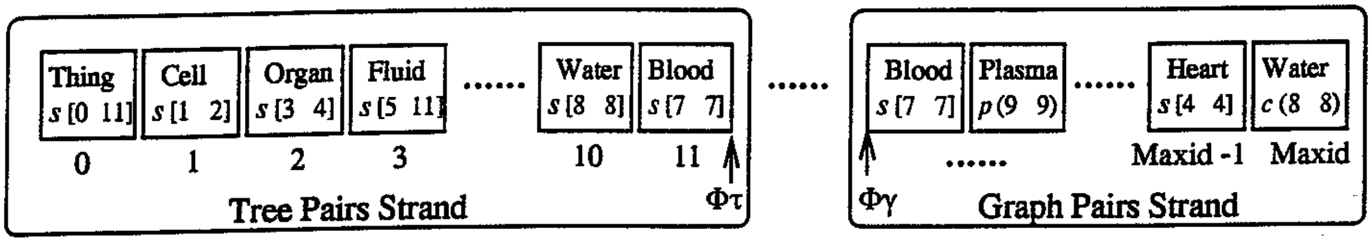


Figure 8: Double Strand Representation

We now introduce some CM-5 terminology. A parallel variable (*pvar*) is an array (possibly multi-dimensional) where every component is maintained by its own processor and all values are usually changed in the same way and in parallel (TMC 1988).

In the algorithm, variables marked with !! are parallel variables, and operations marked with !! or involving parallel variables are parallel operations. The parallel variable pre!! contains for every node its preorder number, and the expression max!! stands for a parallel variable that contains for every node its maximum number, and the expression reltype!! stands for a parallel variable that contains for every number pair its relation type. The functions prenum(), maxnum(), and tree-pair() retrieve the preorder number, maximum number, and the tree pair, respectively, for the given argument.

Additionally, the variable Φ_γ represents the lower bound of the graph pairs strand and Φ_τ represents the upper bound of the tree pairs strand. The parallel function *self-address!!* returns IDs of all active processors and *oddp!!* contains TRUE on a processor if the processor's ID is an odd number. The parallel control structure ACTIVATE-PROCESSORS-WITH consists of two parts. The first part describes a set of processors to be activated. The second part, starting with DO, describes what operations should be performed on all active processors.

Algorithm: Pure Transitivity with IS-A (σ, τ)

- Activate every processor that contains a pair with the IS-A relation type (s).
- (Case 1: a tree path from σ to τ)
  Among active processors, check whether the tree pair of σ is contained in or equal to the tree pair of τ.
- (Case 2: a graph path from σ to τ)
  Among active processors, check whether any processor has a tree pair of τ or a pair of a tree successor of τ at an odd processor ID = x, and the processor with ID = x + 1 contains a pair [propagated] from the tree predecessor of σ.
- Iff Case 1 or Case 2 is the case, return "yes."

We now show a function IS-A-VERIFY that performs pure subclass verification. As we mentioned above, if τ is a tree predecessor of σ (by IS-A-VERIFY-1) or τ is a graph predecessor of σ (by IS-A-VERIFY-2), then σ IS-A τ. Note that as every tree pair has associated with it a single relation type s, it is not necessary to check the relation type for IS-A-VERIFY-1.

```
; B is-a A iff IS-A-VERIFY returns TRUE.
IS-A-VERIFY ($\sigma$, $\tau$: Node): BOOLEAN
  return(IS-A-VERIFY-1($\sigma$, $\tau$) OR
         IS-A-VERIFY-2($\sigma$, $\tau$))


IS-A-VERIFY-1 ($\sigma$, $\tau$: Node): BOOLEAN
  ; If $\tau$ is a tree predecessor of $\sigma$, then
  ; the tree pair of $\tau$ subsumes the tree
  ; pair of $\sigma$.
  ACTIVATE-PROCESSORS-WITH
      prenum(tree-pair($\sigma$)) $\geq$!!
          prenum(tree-pair($\tau$)) AND!!
      maxnum(tree-pair($\sigma$))$\leq$!!
          maxnum(tree-pair($\tau$)) AND!!
      self-address!!() $\leq$!! $\Phi_\tau$
  DO BEGIN
    IF any processor is still active THEN
      return TRUE
  END
```

Now we will show how to verify that σ IS-A τ when τ is a graph predecessor of σ. Remember that a pair of processors (U, V) in the graph pairs strand is used to represent a graph pair propagation. The tree pair in the odd processor (U) is used to represent a node τ and the graph pair in the even processor (V) is used to represent a node which propagates its tree pair to τ. Therefore, we are looking for a pair of processors (U, V) such that the tree pair of τ or of one of its tree successors is contained in processor U and the graph pair of σ or its tree predecessor is contained in processor V. In the following functions the expression *mark*!![x] := y means that the pvar *mark*!! on the processor with the ID x is assigned the value y. We omit the initialization of *mark!!*.

```
IS-A-VERIFY-2 ($\sigma$, $\tau$: Node): BOOLEAN
  ; Activate every occurrence of the tree pair of
  ; $\tau$ or its tree successors associated with the
  ; $s$ relation type in the graph pairs strand. Set
  ; the parallel flag mark!! on the right neighbor
  ; processors of the active processors.
  ACTIVATE-PROCESSORS-WITH
      reltype!! =!! $s$ AND!!
      pre!! $\geq$!! prenum(tree-pair($\tau$)) AND!!
      max!! $\leq$!! maxnum(tree-pair($\tau$)) AND!!
      self-address!!() $\geq$!! $\Phi_\gamma$ AND!!
      oddp!! (self-address!!())
  DO BEGIN
      mark!![self-address!!() +!! 1]:= 1
  END

  ; Test whether any marked processor has the tree
  ; pair with the relation type $s$ from $\sigma$ or from
  ; a tree predecessor of $\sigma$, as a graph pair. If this
  ; is the case, return TRUE.
  ACTIVATE-PROCESSORS-WITH
      reltype!! =!! $s$ AND!!
      pre!! $\leq$!! prenum(tree-pair($\sigma$)) AND!!
      max!! $\geq$!! maxnum(tree-pair($\sigma$)) AND!!
      mark!![self-address!!()] =!! 1
  DO BEGIN
      IF any processor is still active THEN
          return TRUE
  END
```

```
MIXED-RELATION-VERIFY
  ($\xi$: Relation Type; $\sigma$, $\tau$: Node): BOOLEAN
  ; Activate every occurrence of the tree pair of $\tau$
  ; and of its tree successors associated with the
  ; relation type $s$. Set the parallel flag mark!!
  ; on the right neighbor processors of the active
  ; processors.
  ACTIVATE-PROCESSORS-WITH
      reltype!! =!! $s$ AND!!
      pre!! $\geq$!! prenum(tree-pair($\tau$)) AND!!
      max!! $\leq$!! maxnum(tree-pair($\tau$)) AND!!
      self-address!!() $\geq$!! $\Phi_\gamma$ AND!!
      oddp!! (self-address!!())
  DO BEGIN
      mark!![self-address!!() +!! 1]:= 1
  END

  ; Test whether any marked processor has the tree
  ; pair from $\sigma$, or from a tree predecessor of $\sigma$,
  ; as a graph pair with the relation type $\xi$. If this
  ; is the case, return TRUE.
  ACTIVATE-PROCESSORS-WITH
      reltype!! =!! $\xi$ AND!!
      pre!! $\leq$!! prenum(tree-pair($\sigma$)) AND!!
      max!! $\geq$!! maxnum(tree-pair($\sigma$)) AND!!
      mark!![self-address!!()] =!! 1
  DO BEGIN
      IF any processor is still active THEN
          return TRUE
  END
```

Consider again our example of pure transitivity in Figure 6: Is Water a Fluid? The tree pair of Fluid $s[5\ \ 11]$ contains the tree pair of Water $s[8\ \ 11]$. The answer "yes" can be given by comparing these two tree pairs (by IS-A-VERIFY). What about the mixed transitivity example? Is Water contained in Organ? As the query is about Contained-in (c) and σ is Water and τ is Organ, the procedure MIXED-RELATION-VERIFY will be invoked with a list of arguments (c, Water, Organ). We are first looking for Organ and its tree successors. These are nodes with tree pairs contained in $s[3\ \ 4]$. However, as we are interested in propagations, we are looking for these tree successors (or Organ itself) in the graph pairs strand. There we find Heart $s[4\ \ 4]$ with a right neighbor Water $c(8\ \ 8)$ (Figure 8). In the second stage we are looking for a tree predecessors of Water $s[8\ \ 8]$ (or Water itself), but with s replaced by the value of ξ, which is c. This perfectly matches the pair $c(8\ \ 8)$ identified in the first step, and we can conclude that the answer is "yes, Water is contained in Organ." Due to parallel processing, the mixed transitive closure query can be answered in constant time.

An analysis of the parallel operations involved shows that both kinds of queries can be answered with our parallel representation in constant time, independent of the size of the knowledge base (assuming constant machine size) (Wang 1989).

5 EXPERIMENTS WITH InterMED

We have performed a set of experiments that analyze the run-times for purely transitive IS-A queries and mixed relational queries using data from the InterMED. For this experiment, we have used 2495 nodes, 3372 IS-A links, and 682 Pharmaceutic-component-of links from the InterMED. Note that the

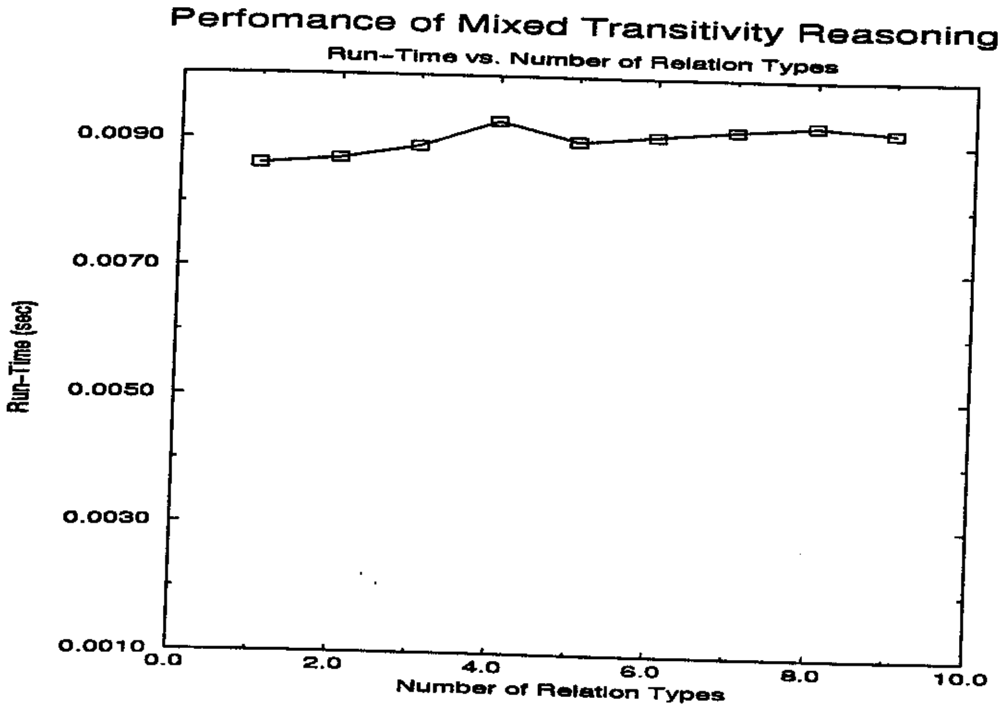


Figure 9: An Experiment of Mixed Transitivity Reasoning

fan-in and fan-out of the hierarchy are not of immediate experimental importance because the node set representation eliminates the explicit IS-A links. The only relevant factor is the number of graph pairs generated. The experiments were performed on a Connection Machine CM-5 (TMC 1988) programmed in *LISP. For this purpose, we constructed a hierarchy for each experiment as follows: (1) We created an IS-A hierarchy (SubClass Hierarchy, SCH); (2) We created a mixed relational hierarchy (Mixed Relational Hierarchy, MRH).

The results in Table 1 show that the run-times for transitive reasoning in both hierarchies are the same within unit processor space (8K virtual processors). We show run-times for "normal" transitive reasoning in the SCH hierarchy, for pure transitivity reasoning in the MRH hierarchy, and for mixed transitivity reasoning. Specifically, IS-A-Verify-1 and IS-A-Verify-2 represent the first and the second cases of the parallel pure transitivity reasoning algorithm in Section 4.

As another experiment, we measured how the number of relations is related to the run-time of mixed relational queries. For this experiment, we tested the run-times by increasing the number of relations in transitive closure queries within constant processor space (8K). These additional relation types were created by a random generator. Figure 9 shows that the run-time for mixed relational transitivity reasoning is independent of the number of relations in a hierarchy. In summary, we can conclude that our mixed transitive queries can be executed in constant time, as in a pure IS-A hierarchy.

6 CONCLUSIONS

In this paper, we have discussed techniques for fast evaluation of transitive queries in mixed relational hierarchies. We have introduced a paradigm based on number pair propagation with a relation type. This paradigm avoids any invalid conclusions of mixed relational transitivity and integrates several relations when needed. Due to our new mixed relational representation and parallel processing, it is possible to perform fast mixed transitivity reasoning. Experimental results using an existing medical vocabulary show that mixed transitivity reasoning can be executed in constant time assuming constant processor space.

Acknowledgements

This work was conducted using the computational resources of the Northeast Parallel Architectures Center (NPAC) at Syracuse University, which is funded

Table 1: Experimental Results for Three Approaches

Hierarchy Type	*Reasoning Type*		*Run-Time*
SCH	*Single Transitivity Reasoning*	*IS-A-Verify-1*	0.00042 (sec)
		IS-A-Verify-2	0.0082 (sec)
MRH	*Pure Transitivity Reasoning*	*IS-A-Verify-1*	0.00042 (sec)
		IS-A-Verify-2	0.0084 (sec)
	Mixed Transitivity Reasoning		0.0086 (sec)

by and operates under contract to DARPA and the Air Force Systems Command, Rome Air Development Center (RADC), Griffiss Air Force Base, NY, under contract# F306002-88-C-0031.

This research was (partially) done under a cooperative agreement between the National Institute of Standards and Technology Advanced Technology Program (under the HIIT contract, number 70NANB5H1011) and the Healthcare Open Systems and Trials, Inc consortium.

References

R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. In *Proceedings of the 1989 ACM SIGMOD International Conference on the Management of Data*, pages 253–262, Portland, OR, 1989.

R. Agrawal, S. Dar, and H. V. Jagadish. Direct transitive closure algorithms: Design and performance evaluation. *ACM Transactions on Database Systems*, 15(3):428–458, 1990.

J. J. Cimino, A. A. Aguirre, S. B. Johnson, and P. Peng. Generic queries for meeting clinical information needs. *Bulletin of the Medical Library Association*, 81(2):195–206, 1993.

J. J. Cimino, P. D. Clayton, G. Hripcsak, and S. B. Johnson. Knowledge-based approaches to the maintenance of a large controlled medical terminology. *Journal of the American Medical Informatics Association*, 1(1):35–50, 1994.

J. J. Cimino, P. L. Elkin, and G. O. Barnett. As we may think: The concept space and medical hypertext. *Computers and Biomedical Research*, 25:238–263, 1992.

J. J. Cimino, G. Hripcsak, S. B. Johnson, and P. D. Clayton. Designing an introspective, multipurpose, controlled medical vocabulary. In *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care*, pages 513–518. IEEE Computer Society Press, Los Alamitos, CA, 1989.

Paul R. Cohen and Cynthia L. Loiselle. Beyond isa: Structures for plausible inference in semantic networks. In *Proceedings of AAAI-88*, pages 415–420, Saint Paul, Minnesota, 1988.

M. P. Evett, W. A. Andersen, and J. A. Hendler. Massively parallel support for efficient knowledge representation. In *Proc. of the 13th Int. Joint Conference on Artificial Intelligence*, pages 1325–1330. Morgan Kaufmann, San Mateo, CA, 1993.

M. P. Evett, J. A. Hendler, and W. A. Andersen. Massively parallel support for computationally effective recognition queries. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 297–302. MIT Press, Cambridge, MA, 1993.

M. Evett, L. Spector, and J. Hendler. Knowledge representation on the connection machine. In *Supercomputing '89*, pages 283–293, Reno, Nevada, 1989.

J. Geller and C. Y. Du. Parallel implementation of a class reasoner. *Journal of Experimental and Theoretical Artificial Intelligence*, 3:109–127, 1991.

J. Geller. Upward-inductive inheritance and constant time downward inheritance in massively parallel knowledge representation. In *Proceedings of the Workshop on Parallel Processing for AI at IJCAI 1991*, pages 63–68, Sydney, Australia, 1991.

J. Geller. Innovative applications of massive parallelism. *AAAI 1993 Spring Symposium Series Reports, AI Magazine*, 14(3):36, 1993.

J. Geller. Massively parallel knowledge representation. In *AAAI Spring Symposium Series Working Notes: Innovative Applications of Massive Parallelism*, pages 90–97, 1993.

J. Geller. Advanced update operations in massively parallel knowledge representation. In H. Kitano and J. Hendler, editors, *Massively Parallel Artificial Intelligence.* AAAI/MIT Press, 1994, forthcoming.

J. Geller. Inheritance operations in massively parallel knowledge representation. In L. Kanal, V. Kumar, H. Kitano, and C. Suttner, editors, *Parallel Processing for Artificial Intelligence.* Elsevier Science Publishers, Amsterdam, 1994, forthcoming.

M. Halper, J. Geller, and Y. Perl. An OODB "part" relationship model. In Yelena Yesha, editor, *Proceedings of the First International Conference on Information and Knowledge Management*, pages 602–611, Baltimore, MD, nov 1992.

M. Halper. *A Comprehensive Part Model and Graphical Schema Representation for Object-Oriented Databases*. PhD dissertation, CIS Department, New Jersey Institute of Technology, 1993.

M. Halper, J. Geller, and Y. Perl. Value propagation in object-oriented database part hierarchies. In *Proceedings of the 2nd Int'l Conference on Information and Knowledge Management*, pages 606–614. Washington, DC, 1993.

M. Halper, J. Geller, and W. Klas. Integrating a part relationship into an open oodb system using metaclasses. In *Proceedings of the 3rd Int'l Conference on Information and Knowledge Management*, pages 10–17, Gaithersburg, MD, 1994.

James Hendler, Jaime Carbonell, Douglas Lenat, Richiro Mizoguchi, and Paul Rosenbloom. Very large knowledge bases–architecture vs engineering. In *Proceedings of IJCAI 1995*, pages 2033–2037, Montreal, Quebec, 1995.

H. Kitano and T. Higuchi. High performance memory-based translation on IXM2 massively parallel associative memory processor. In *Proceedings of IJCAI 1991*, pages 149–154, Sydney, Australia, 1991.

H. Kitano and T. Higuchi. Massively parallel memory-based parsing. In *Proceedings of IJCAI 1991*, pages 918–924, Sydney, Australia, 1991.

H. Kitano. Massively parallel AI and its application to natural language processing. In *Proceedings of the Workshop on Parallel Processing for AI at IJCAI 1991*, pages 99–105, Sydney, Australia, 1991.

H. Kitano, D. Moldovan, and S. Cha. High performance natural language processing on semantic network array processor. In *Proceedings of IJCAI 1991*, pages 911–917, Sydney, Australia, 1991.

E. Y. Lee and J. Geller. Representing transitive relationships with parallel node sets. In Bharat Bhargava, editor, In *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, pages 140–145. IEEE Computer Society Press, Los Alamitos, CA, 1993.

E. Y. Lee and J. Geller. Parallel operations on class hierarchies with double strand representations. In *IJCAI-95 Workshop Program Working Notes*, pages 132–142, Montreal, Quebec, 1995.

E. Y. Lee and J. Geller. Constant time inheritance with parallel tree covers. In *Proceedings of the Ninth Florida AI Research Symposium*, pages 243–250, Key West, FL, 1996.

E. Y. Lee. Massively parallel reasoning in transitive relationship hierarchies *PhD dissertation*, CIS Department, New Jersey Institute of Technology, 1996.

M. R. Quillian. Semantic memory. In M. L. Minsky, editor, *Semantic Information Processing*, pages 227–270. The MIT Press, Cambridge, MA, 1968.

L. Shastri and V. Ajjanagadde. An optimally efficient limited inference system. In *Proceedings of IJCAI-90*, pages 563–570, Boston, MA, 1990.

L. K. Schubert. Problems with parts. In *Proc. of the 6th International Joint Conference on Artificial Intelligence*, pages 778–784. Morgan Kaufmann Publishers, Los Altos, CA, 1979.

L. Shastri. *Semantic Networks: an Evidential Formalization and its Connectionist Realization*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

L. Shastri. Default reasoning in semantic networks: a formalization of recognition and inheritance. *Artificial Intelligence*, 39(3):283–356, 1989.

L. K. Schubert, M. A. Papalaskaris, and J. Taugher. Determining type part, color, and time relationships. *Computer*, 16(10):53–60, 1983.

L. K. Schubert, M. A. Papalaskaris, and J. Taugher. Accelerating deductive inference: special methods for taxonomies colors and times. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier*, pages 187–220. Springer Verlag, New York, NY, 1987.

Thinking Machines Corporation. **LISP Reference Manual Version 5.0 edition*. Thinking Machines Corporation, Cambridge, MA, 1988.

D. L. Waltz. Massively parallel AI. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1117–1122. Morgan Kaufmann, San Mateo, CA, 1990.

W. Wang, S. Iyengar, and L. M. Patnaik. Memory-based reasoning approach for pattern recognition of binary images. *Pattern Recognition*, vol. 22, no 5, pp. 505–518, 1989.

M. E. Winston, R. Chaffin, and D. Herrmann. A taxonomy of part-whole relations. *Cognitive Science*, 11(4):417–444, 1987.