Mechanisms for Reasoning about Sets

Michael P. Wellman*
MIT Lab for Computer Science
545 Technology Square
Cambridge, MA 02139
mpw@Zermatt.LCS.MIT.EDU

Reid G. Simmons†
MIT AI Laboratory
545 Technology Square
Cambridge, MA 02139
reid@OZ.AI.MIT.EDU

Abstract

The SEt Reasoning Facility (SERF) integrates mechanisms for propagating membership propositions, deriving relations between sets, and reasoning about closure and cardinality into an efficient utility package for reasoning about sets. Assertions about relations between sets are compiled into a constraint network defined entirely in terms of union, complement, and emptiness constraints. The constraint network supports multiple modes of inference, such as local propagation of membership propositions and graph search for set relations using a transitivity table. SERF permits closure assertions of the form "all members of set S are known" and utilizes this capability to permit selective applications of closed-world assumptions. Cardinality constraints are handled by a general quantity reasoner. An example from geologic interpretation illustrates the value of mutually constraining sources of information in a typical application of reasoning about sets in commonsense problem-solving.

1 Introduction

Sets play an important role in representing and reasoning about the commonsense world. Many attributes of real-world objects are naturally represented as sets, for instance, the set of objects on top of a table, the set of rock formations along the surface of the Earth, and the set of parents a person has. Reasoning about such attributes, especially about the changes that occur to them, requires mechanisms for reasoning about:

1. *Relationships between sets.* If the set of green blocks is disjoint from the set of blocks in the room and the blocks on the table are a subset of the blocks in the room, then there are no green blocks on the table.
2. *Combinations of sets.* After erosion occurs, the set of geologic formations on the surface of the Earth is the union of the newly exposed formations with the initial surface formations minus those eroded away.
3. *Elements of sets.* If we know that Joe and Amy are biological parents of John, then George cannot be a parent of John. If Mary's parents are Amy and Roy, then John and Mary are step-siblings.

*Current address: AFWAL/AAI, Wright-Patterson AFB, OH 45433.

†Current address: Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

We have implemented mechanisms to support these and other tasks, integrating them into the SEt Reasoning Facility (SERF). SERF records facts about sets of interest and answers queries as directed by the user or problem-solver. SERF is designed to derive propositional facts about particular sets—not to prove theorems about properties of sets in general (contrast with ONTIC [McAllester, 1987]). By keeping all reasoning local and vivid (limited use of disjunction and negation), we gain efficiency in doing common set-related inferences, at the cost of completeness and generality.

A powerful feature of SERF is its integration of different types of information, in particular ordinal relationships (such as $\subseteq$) and set membership. The various types of information are mutually constraining, for instance, SERF computes ordinal relationships from knowledge about membership and vice versa. SERF draws relatively weak conclusions when little information is known about the members of a set but gives more precise answers as more detailed information becomes available. For example, knowing only that $C = A \cup B$ we can infer that $|C| \leq |A| + |B|$, but given all the members of A and B we can determine the exact membership and cardinality of C.

The following section illustrates some of SERF's capabilities with two example applications. The remaining sections describe the representations and algorithms employed to achieve these results. Section 3 introduces the constraint network representation of set operations and describes the mechanisms for propagating membership propositions. Facilities for representing and deriving relationships between sets are presented in Section 4. Section 5 discusses SERF's closure mechanisms: techniques for asserting that a set's elements are exactly those that are known members. Reasoning about cardinality is the subject of Section 6.

2 Applications: Two Examples

Reasoning about sets is important in simulating and interpreting physical situations [Simmons and Davis, 1987]. For example, in interpreting the sequence of events that could form a geologic region, one must often reason about how the set of rock formations along the surface of the Earth change as a result of the action of geologic events, such as erosion and deposition.

The effect of erosion on the set of formations along the Earth's surface can be represented by the equation $S_2 = (S_1 - TE) \cup EX$, where S_1 is the set of formations on the surface before erosion, S_2 is the set after erosion, TE is the set of formations totally eroded away, and EX is the set of newly exposed formations that were under S_1 (see Figure 1). In addition, we know that TE is a subset of S_1

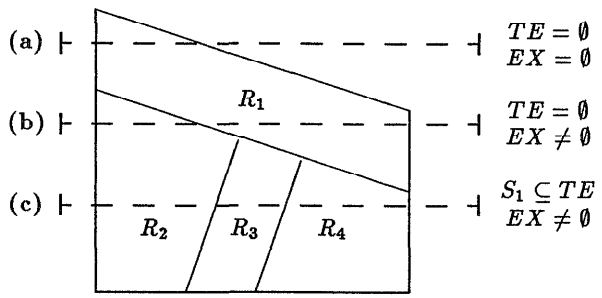

Figure 1: Geologic interpretation example. The dashed lines represent hypothetical erosion patterns.

and EX is disjoint from S_1.

In interpreting a geologic region, we are often interested in the relationships between the various sets of rock formations, such as between S_2 and S_1, the new and old surfaces, and between S_2 and the underlying rocks EX. From the above description of erosion, SERF can infer that S_2 is a superset of EX and that EX and TE are disjoint. That little else can be derived is to be expected since the general description indicates nothing about the extent of erosion. As we add more constraints, however, SERF infers more detailed relationships. For example, if we assert that TE and EX are both empty (Figure 1, case a), SERF infers that S_2 is equal to S_1 and disjoint from EX. When we assert that TE is empty but EX is not (case b – rocks are partially eroded, exposing some underlying formations), SERF infers that S_2 is a proper superset of both S_1 and EX. Finally, asserting that $S_1 \subseteq TE$ (case c – all formations currently on the surface are eroded away) enables SERF to infer that $S_2 = EX$.

Alternatively, SERF can reach these conclusions using constraints on the membership of sets. If, in conjunction with the erosion equation $S_2 = (S_1 - TE) \cup EX$, we assert that R_1 is the only member of S_1, that R_2, R_3, and R_4 are the only members of EX and that R_1 is a member of TE, SERF will conclude that $S_2 = \{R_2, R_3, R_4\}$ and thus is equal to EX and disjoint from S_1.

We have also applied set membership reasoning to the problem of unifying terms involving set variables. URP, a program for reasoning about preferences represented as utility functions [Wellman, 1985], performs goal-directed inference from a collection of utility decomposition proof rules similar to the following:

$$(A, B \subseteq C) \wedge (A \cap B \neq \emptyset) \wedge UI(A, C - A) \wedge UI(B, C - B) \vdash GUI(A - B, C - (A - B)).$$

For our current purposes, it is sufficient to note that UI and GUI are predicates of multiattribute utility theory describing the permissible preference interactions among sets of utility attributes. Given a goal formula, such as $GUI(\{x_1, x_2\}, \{x_3, x_4, x_5\})$, the unification problem is to find values of A, B, and C to instantiate the premise. To help reduce the combinatorial search required to find unifiers, SERF is used to constrain the members assigned to set terms. For example, after URP binds $A - B$ to $\{x_1, x_2\}$ and $C - (A - B)$ to $\{x_3, x_4, x_5\}$, SERF determines that $C = \{x_1, \ldots, x_5\}$ and that x_1 and x_2 must be contained in A but not in B.

3 Set Constraint Networks

SERF represents assertions about sets in a constraint network [Sussman and Steele, 1980]. Nodes in the network are set objects, encoding such information as the elements that are known to be members and bounds on the set's cardinality. Constraint links enforce relations among the sets they connect.

Each set is associated with four types of information:

1. Propositions about membership of various elements in the set, of the form $x \in A$.
2. Whether the set is empty.
3. Whether the set is closed, that is, all of its members are known.
4. Cardinality of the set.

All set-related propositions, are recorded in a truth maintenance system (TMS) [McAllester, 1980] to provide for dependency-directed updating upon addition and deletion of assertions. Propositions are marked **true**, **false**, or **unknown** and are tagged with a justification for that labeling.

The two primitive set operations supported in SERF's constraint network representation are **union** and **complement**. These are sufficient to represent the standard boolean set operations. For example, the intersection operation $A \cap B$ can be rendered in terms of our primitives by $\overline{\bar{A} \cup \bar{B}}$, where $\bar{S}$ denotes the complement of a set S (see Figure 2).

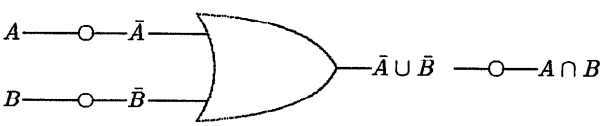

Figure 2: A constraint network representing the intersection of A and B, built from a **union** constraint (the OR gate) and three **complement** constraints (the "inverter" circles).

The constraint network is used to propagate assertions about set membership. Given the proposition $x \in A$ (or its negation, $x \notin A$), the constraints determine whether x is an element of sets related to A. The **complement** constraint ensures the equivalence of $x \in A$ and $x \notin \bar{A}$ using the following two disjunctive clauses:

$$x \in A \vee x \in \bar{A} \quad (1)$$

$$x \notin A \vee x \notin \bar{A}. \quad (2)$$

The **union** constraint for $A \cup B$ is represented by three propositional clauses:

$$x \notin A \vee x \in A \cup B, \quad (3)$$

$$x \notin B \vee x \in A \cup B, \quad (4)$$

$$x \in A \vee x \in B \vee x \notin A \cup B. \quad (5)$$

Whenever all but one disjunct in a clause is marked **false**, the remaining proposition is declared **true**. In Figure 2, for example, asserting $x \notin A$ implies that $x \in \bar{A}$ by (1), which implies $x \in \bar{A} \cup \bar{B}$, by (3). This in turn implies that $x \notin A \cap B$, by (2). We can show that our canonization of

set operations into **union** and **complement** constraints preserves the membership inferences derivable from a direct implementation of the boolean set operations.

SERF's membership reasoning is incomplete, however, in part due to the locality of constraint propagation. Suppose, for example, we assert that $x \in B \cup C$, $x \notin A$, and that sets $A \cup B$ and $A \cup C$ are equal. A global analysis of the constraints reveals that $x \in B$ and $x \in C$, since all elements not in A must be in both or neither of B and C. This conclusion does not follow, however, by considering each constraint individually.

The assertion language is limited in its ability to express disjunction, negation, and quantification over sets. For instance, we cannot encode directly such disjunctive membership assertions as "$x \in A \vee y \in A$." Although such a condition may be implied by the network as a whole in that assertions that one is a non-member will result in the other being declared a member, it cannot be encoded in the set node A itself. For example, we can encode "$x \in A \vee x \in B$" by asserting $x \in A \cup B$. Then from $x \notin A$ SERF can infer that $x \in B$.

4 Relations Between Sets

SERF uses the same constraint networks to encode relations between sets and to infer new set relationships. For example, we can assert that one set is a subset of another, or try to deduce whether two sets are disjoint. By using the same representation for reasoning about both membership and relations, SERF exploits the mutual constraint between the two types of information.

4.1 Basic Relations

Our inference mechanisms support the four binary set relationships: subset ($\subseteq$), superset ($\supseteq$), disjoint ($\|$), and total (T)[1] and their respective negations: $\not\subseteq$, $\not\supseteq$, $\not\|$, and $\not T$. Table 1 presents their definitions in terms of set membership. Equality is represented as the conjunction of $\subseteq$ and $\supseteq$.

R	Definition of $A\ R\ B$	Definition of $A\ \not R\ B$
$\subseteq$	$\forall x.x \notin A \vee x \in B$	$\exists x.x \in A \wedge x \notin B$
$\supseteq$	$\forall x.x \in A \vee x \notin B$	$\exists x.x \notin A \wedge x \in B$
$\|$	$\forall x.x \notin A \vee x \notin B$	$\exists x.x \in A \wedge x \in B$
T	$\forall x.x \in A \vee x \in B$	$\exists x.x \notin A \wedge x \notin B$

Table 1: The basic binary set relations.

In order to integrate knowledge about set relations with the membership reasoning mechanisms, SERF compiles relation assertions into networks of **union** and **complement** constraints augmented by assertions about the emptiness of sets. For example, SERF translates $A \| B$ into an assertion that the set $\overline{\bar{A} \cup \bar{B}}$ (the intersection of A and B) is empty. Using the membership proposition clauses of Section 3 in conjunction with the knowledge that nothing is a member of the empty set, SERF enforces the constraint that members of A are not members of B, and vice versa. If we retract the disjoint assertion (by retracting the emptiness constraint), SERF automatically withdraws support from any membership propositions derived in this manner. Table 2 lists the constraint representations of the eight basic relations.

R	constraint	R	constraint
$\subseteq$	$empty(\overline{\bar{A} \cup B})$	$\not\subseteq$	$nonempty(\overline{\bar{A} \cup B})$
$\supseteq$	$empty(\overline{A \cup \bar{B}})$	$\not\supseteq$	$nonempty(\overline{A \cup \bar{B}})$
$\|$	$empty(\overline{\bar{A} \cup \bar{B}})$	$\not\|$	$nonempty(\overline{\bar{A} \cup \bar{B}})$
T	$empty(\overline{A \cup B})$	$\not T$	$nonempty(\overline{A \cup B})$

Table 2: Constraint network representations of the eight basic relations.

4.2 Deriving Relations via Path Search

Answering queries about the relations holding between sets is an important set reasoning task. SERF derives set relations by composing paths of relations in the constraint network using the transitivity of relations. For example, if A is disjoint from B and B is a superset of C, A must be disjoint from C as well. The implication of $(A\ R_1\ B) \wedge (B\ R_2\ C)$ is $A\ (R_1 \circ R_2)\ C$, where $R_1 \circ R_2$ is the relation, if any, in the cell of Table 3 corresponding to the row for R_1 and the column for R_2.

$\circ$	$\subseteq$	$\not\subseteq$	$\supseteq$	$\not\supseteq$	$\|$	$\not\|$	T	$\not T$
$\subseteq$	$\subseteq$			$\not\supseteq$	$\|$			$\not T$
$\not\subseteq$			$\not\subseteq$				$\not\|$	
$\supseteq$		$\not\subseteq$	$\supseteq$			$\not\|$	T	
$\not\supseteq$	$\not\supseteq$				$\not T$			
$\|$		T	$\|$			$\not\supseteq$	$\subseteq$	
$\not\|$	$\not\|$			$\not\subseteq$				
T	T			$\not\|$	$\supseteq$			$\not\subseteq$
$\not T$			$\not T$				$\not\supseteq$	

Table 3: The set relation transitivity table.

The table is complete for chains in the following sense: if all we know about sets $S_1 \ldots S_n$ is the chain of relations $S_i\ R_i\ S_{i+1}$, with $i = 1, \ldots, n-1$, then $R_1 \circ \cdots \circ R_n$ is the strongest implied basic relation. In addition, the transitivity operator ($\circ$) is associative, so relation pairs in the chains may be combined in any order.

To determine the relations holding between a pair of sets, SERF searches the constraint network, combining the relations found on different paths. The search proceeds in a breadth-first manner, maintaining at each node the set of basic relations known to hold with the starting set. Search proceeds from a node only if this collection of relations has been strengthened on the incoming path. The method is similar to that employed by others for deriving temporal and arithmetic relations by transitivity (e.g., [Allen, 1983; Simmons, 1986]).

For example, in the simple relation network of Figure 3, the derived relation between A and D is the conjunction of those found on the two paths: $T \circ \not\supseteq = \not\|$ and $\supseteq \circ T = T$.

[1] $A\ T\ B$ means that together A and B span the universe of objects.

Combining each of these with $D \parallel E$ yields $A \not\subseteq E$ and $A \supseteq E$, that is, A is a proper superset of E. In our path search algorithm, each set node can be visited at most four times, once for each of the basic relations. Because the algorithm adds no new structure to the constraint network, its worst-case complexity is $O(r)$, where r is the number of relations asserted between sets.

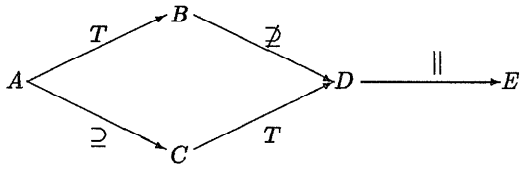

Figure 3: A network of relations. Path search using the transitivity table reveals that $A \supset E$.

As described above, SERF encodes the relations of Table 1 using only **union**, **complement**, and emptiness constraints. Thus, in searching this network, SERF must first translate **union** and **complement** relations into the corresponding relations of Table 3. A **complement** constraint expands into $A \parallel \bar{A}$ and $A\ T\ \bar{A}$. A **union** constraint implies that both A and B are subsets of $A \cup B$. Degenerate sets gain some relations automatically: $\emptyset$ is a disjoint subset of any set, and $\bar{\emptyset}$ is a total superset. Non-degeneracy constraints also restrict the possible combinations of relations that can hold between sets. SERF enforces the following constraints:

$$A \neq \emptyset \Rightarrow [A \not\subseteq B \vee A \not\parallel B],$$

$$A \neq \bar{\emptyset} \Rightarrow [A \not\supseteq B \vee A \not T B].$$

For example, if A is nonempty and $A \subseteq B$, SERF deduces that $A \not\parallel B$ and uses that relation in its path search algorithm.

4.3 Deriving Relations via Membership Comparison

SERF also derives relations between sets by comparing their members. This mechanism enables SERF to deduce relationships even between sets that are not connected in the constraint network. For example, if A is known to contain elements x_1, x_2, and x_3 and B contains x_2, x_4, and x_5, SERF concludes that A and B are not disjoint, since they have an element in common. In addition, if $x_3 \in \bar{C}$, SERF infers that $A \not\subseteq C$, since one of the known elements of A is not an element of C.

The necessary conditions for membership comparison can be easily derived from Table 1. For example, the definition of subset entails that $A \subseteq B$ if all the elements of A are elements of B or, conversely, if all the elements of $\bar{B}$ are elements of $\bar{A}$. Similarly, $A \not\subseteq B$ if some element of A is an element of $\bar{B}$ or, equivalently, some element of $\bar{B}$ is an element of A. Table 4 presents the complete set of conditions needed to derive relationships by membership comparison. In the table, *some* means that at least one of the elements of the first set is a member of the second set. *All* means that all of the known members of the first set are members of the second *and* that the first set is closed,

Relationships between set A and set B

Comparison	Some	All	Comparison	Some	All
A with B	$\not\parallel$	$\subseteq$	B with A	$\not\parallel$	$\supseteq$
A with $\bar{B}$	$\not\subseteq$	$\parallel$	B with $\bar{A}$	$\not\supseteq$	$\parallel$
$\bar{A}$ with B	$\not\supseteq$	T	$\bar{B}$ with A	$\not\subseteq$	T
$\bar{A}$ with $\bar{B}$	T	$\supseteq$	$\bar{B}$ with $\bar{A}$	T	$\subseteq$

Table 4: Using membership comparison to derive ordinal relationships.

that is, the set has no members other than those explicitly enumerated.

Performing comparison by sorting the membership lists and then iterating, the computational complexity of this algorithm is $O(n \log n)$, where n is the number of known elements in the sets. In the problems we have encountered, the membership comparison mechanism is more efficient than the path search mechanism since the number of set members are typically much less than the number of set relations in the network. Hence, our strategy is to use membership comparison first and try path search only if more information remains to be derived.

5 Closure

Asserting that a set is closed means that the only members of the set are those currently known to the system. The knowledge that a set is closed adds significantly to the range of inferences SERF can perform. For example, in comparing the members between two sets (see Section 4.3), SERF cannot determine relations such as subset or disjoint unless it is known that one of the sets is closed. Similarly, the membership propagation constraints make use of set closure. If B is closed and x is not known to be a member of B (that is, the proposition $x \in B$ is **false** or **unknown**), SERF infers that $x \in \bar{B}$. (This inference relies on an implicit SERF assumption that all distinct terms denote distinct objects.)

Defining closure in terms of the current state of knowledge complicates dependency maintenance. The difficulty appears in the following situation: suppose we assert that $x_1 \in A$ and that A is closed. If we issue a query about x_2, the system will respond that $x_2 \in \bar{A}$, justified by the assertions that A is closed and $x_1 \in A$. If we then retract the assertion that A is closed, the TMS retracts the assertion that $x_2 \in \bar{A}$. Thus, subsequently asserting $x_2 \in A$ causes no contradiction. If we reassert that A is closed, however, we do not want the TMS to reassert that $x_2 \in \bar{A}$, since that would conflict with the assertion that $x_2 \in A$.

To guard against such unwanted contradictions, SERF implements the closure assertion as "the set S has exactly n members." When a closure assertion is made, SERF counts the number of currently known elements and creates a closure assertion of this form. The assertion is justified by the current membership propositions of the set, so that the closure is retracted if any of the membership propositions are retracted. In the problem above, the first assertion of closure becomes "A has exactly one member," justified by $x_1 \in A$, and the second "A has exactly two members," justified by both memberships. Since $x_2 \in \bar{A}$ is justified

by the first assertion, closing A the second time will not result in a contradiction.

Selective application of closed-world assumptions is a useful technique in commonsense reasoning. SERF enables users to make closed-world assumptions over the members of sets through a simple extension to the closure mechanism described above. The only difference is that no contradiction is raised if an object is asserted to be a member of a set closed under the closed-world assumption. Whenever an element is added to or removed from such a set, SERF retracts the current closure assumption, modifies the membership propositions, then imposes a new closure assertion.

SERF's mechanism for closed-world assumptions implements a form of non-monotonic reasoning, where conclusions drawn from a given set of premises may become invalidated by subsequent assertions. The underlying TMS itself is monotonic; conclusions are withdrawn only by explicit retraction.

6 Cardinality

SERF provides mechanisms for describing and reasoning about the cardinality of sets. The mechanism uses the Quantity Lattice [Simmons, 1986] to reason about arithmetic relations, addition and subtraction, and numeric interval constraints. The cardinality reasoning mechanism is a separable component of SERF in that none of the mechanisms described above depend on cardinality information. This gives the user the option of not utilizing the cardinality component if the added power (and added computational complexity) is not required.

The cardinality of a set is implemented as a quantity in the Quantity Lattice. The value of a quantity is constrained by its arithmetic relations ($<, \leq, >, \geq, =, \neq$) to other quantities or numbers. Using this mechanism, one can constrain the number of elements in a set without specifying its exact elements.

SERF ensures that the cardinality of a set is consistent with its membership, emptiness, and closure constraints. Whenever a member is added to or removed from a set, an assertion is made that the cardinality is greater than or equal to the number of currently known elements. The assertion that a set is closed implies that the upper and lower bounds on the cardinality of the set are equal. Conversely, if the upper bound on cardinality is constrained to be equal to the number of known elements, the set is asserted to be closed. In all cases, appropriate dependencies are recorded to facilitate retraction and explanation.

Cardinality constraints are propagated across **union** relations. Given $A \cup B$, the system asserts $|A| \leq |A \cup B|$, $|B| \leq |A \cup B|$, $|A \cup B| \leq |A| + |B|$, $|\overline{A \cup B}| \leq |\bar{A}|$, and $|\overline{A \cup B}| \leq |\bar{B}|$. In addition, if the size of the universe is known, the system asserts that the sum of the cardinalities of a set and its complement equals the cardinality of the universal set.

The use of cardinality increases the range of inferences that SERF can perform. For example, if we assert that the size of the set of John's parents is two and assert that Mary and Joe are members of the parent set of John, then SERF can infer that George cannot be a parent of John since that would violate the cardinality constraints.

The system can also use cardinality information to infer new relations. For example, knowing that $|A|$ is greater than $|B|$, the system can infer that $A \not\subseteq B$. Using the definitions of Table 1, the implication

$$[\forall x. x \notin S_1 \vee x \in S_2] \Rightarrow |S_1| \leq |S_2|,$$

and the equivalence between $x \notin S_1$ and $x \in \bar{S}_1$, we can also derive $A \not\supset B$ from $|\bar{A}| > |B|$ and $A \not\parallel B$ from $|A| > |\bar{B}|$. As with our other inference mechanisms, this is an efficient but incomplete means of determining set relations.

7 Summary

SERF is a utility for generic set reasoning that integrates mechanisms for propagating membership propositions, deriving relations between sets, and reasoning about closure and cardinality. The central constraint network mechanism integrates multiple sources of knowledge and supports multiple modes of inference, such as local propagation and path search. We have found a comprehensive set reasoner to be useful in several domains and expect these techniques to be applicable to a wide variety of commonsense reasoning tasks involving sets.

Acknowledgment

Yishai Feldman, Brian Williams, and Alex Yeh contributed helpful comments on an earlier draft. This work was supported by Schlumberger, National Institutes of Health Grant No. R01 LM04493 from the National Library of Medicine, and the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

References

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[McAllester, 1980] David A. McAllester. An outlook on truth maintenance. AIM 551, MIT Artificial Intelligence Laboratory, 1980.

[McAllester, 1987] David A. McAllester. ONTIC: A knowledge representation system for mathematics. TR 979, MIT Artificial Intelligence Laboratory, 1987.

[Simmons, 1986] Reid Simmons. "Commonsense" arithmetic reasoning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 118–124. AAAI, August 1986.

[Simmons and Davis, 1987] Reid Simmons and Randall Davis. Generate, test, and debug: Combining associational rules and causal models. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1071–1078, 1987.

[Sussman and Steele, 1980] Gerald Jay Sussman and Guy Lewis Steele Jr. CONSTRAINTS—A language for expressing almost-hierarchical descriptions. *Artificial Intelligence*, 14:1–39, 1980.

[Wellman, 1985] Michael Paul Wellman. Reasoning about preference models. TR 340, MIT Laboratory for Computer Science, May 1985.