Mechanisms for Commonsense Reasoning about Sets

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

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

March 7, 1988

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. 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 relationships between sets, such as subset and disjoint; combinations of sets, such as union and intersection; and elements of sets, including cardinality and closure.

We have integrated these types of reasoning into the SEt Reasoning Facility (SERF). SERF records facts about the sets of interest and answers queries as directed by the user or problem-solver. A powerful feature of SERF is its integration of knowledge, in particular, ordinal relationships (such as $\subseteq$) and set membership information. 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.

Reasoning about sets is important in simulating and interpreting physical situations [5, 7]. 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 deposition and erosion.

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 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 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_1 is equal to S_2 and disjoint from

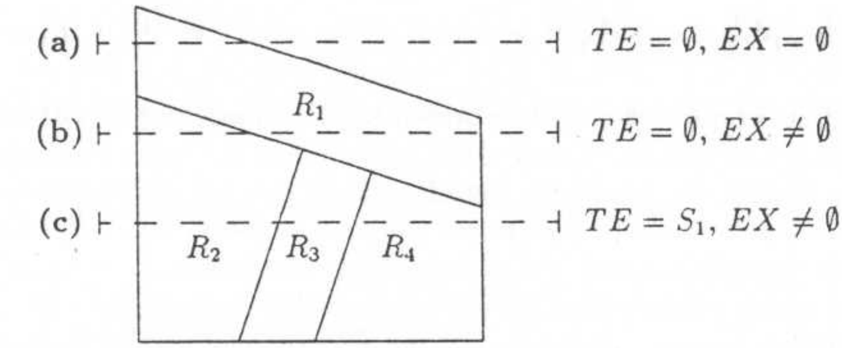

Figure 1: Geologic interpretation example. The dashed lines represent hypothetical erosion patterns.

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) causes 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 general description of erosion given above, 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.

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 [9], performs goal-directed inference from a collection of utility decomposition proof rules similar to the following:

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

$$\vdash GUI(A-B, C-(A-B)).$$

For our current purposes, it is sufficient to note that UI and GUI are utility-theoretic predicates describing the possible 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 identifies $A - B$ with $\{x_1, x_2\}$ and $C - (A - B)$ with $\{x_3, x_4, x_5\}$, 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.

2 Set Constraint Networks

SERF represents assertions about sets in a constraint network [8]. 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 object 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; and 4) cardinality of the set. All facts about sets are recorded in a truth maintenance system (TMS) [2] to provide for dependency-directed updating upon addition and deletion of assertions.

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 according to $\overline{\bar{A} \cup \bar{B}}$, where $\bar{S}$ denotes the complement of a set S (see Figure 2).

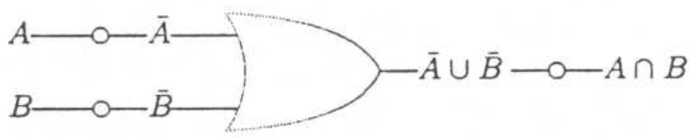

Figure 2: A constraint network representing the intersection of A and B, built from a **union** constraint (the OR gate) and three **complement** constraints (the "inverter" circles).

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 $x \notin \bar{A}$. The **union** constraint encodes that $(x \in A \vee x \in B \iff x \in A \cup B)$. In Figure 2, for example, asserting $x \notin A$ implies that $x \in \bar{A}$, which implies $x \in \bar{A} \cup \bar{B}$, which, in turn, implies that $x \notin A \cap B$. As evidenced by this example, we can show that our canonicalization 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, 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.

3 Relations Between Sets

SERF uses the same constraint networks to encode relations between sets and 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 can exploit the mutual constraint between the two types of information.

Our inference mechanisms support the four binary set relationships: subset ($\subseteq$), superset ($\supseteq$), disjoint ($\|$), and total (T) and their respective negations: $\not\subseteq$, $\not\supseteq$, $\not\|$, and $\not T$. SERF compiles assertions about set relations 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 2 in conjunction with the knowledge that nothing is a member of the empty set, SERF propagates membership in A to membership in B in accordance with the disjointness definition. If we retract the disjoint assertion (by retracting the emptiness constraint), SERF automatically withdraws support from any membership propositions derived in this manner.

3.1 Deriving Relations through Network Search

SERF derives set relations by composing paths of relations in the constraint network using Table 1. For example, if A is disjoint from B and B is a superset of C, A must be disjoint from C as well. To determine the relations holding between a pair of sets, SERF performs a breadth-first search in the constraint network, combining the relations found on different paths. The method is similar to that employed by Allen [1] for deriving temporal relations by transitivity. 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 $\supseteq \circ T = T$. Combining each of these with $\|$ yields $A \not\subseteq E$ and $A \supseteq E$, that is, A is a proper superset of E.

As described above, SERF encodes the basic binary set relations using only **union**, **complement**, and emptiness constraints. In searching this network, SERF computes the local relations by inspection of these constraints. A **complement** constraint expands into $\|$ and T. 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

$\circ$	$\subseteq$	$\not\subseteq$	$\supseteq$	$\not\supseteq$	$\parallel$	$\not\parallel$	T	$\not T$
$\subseteq$	$\subseteq$			$\not\supseteq$	$\parallel$			$\not T$
$\not\subseteq$			$\not\subseteq$				$\not\parallel$	
$\supseteq$		$\not\subseteq$	$\supseteq$			$\not\parallel$	T	
$\not\supseteq$	$\not\supseteq$				$\not T$			
$\parallel$		$\not T$	$\parallel$			$\not\supseteq$	$\subseteq$	
$\not\parallel$	$\not\parallel$				$\not\subseteq$			
T	T			$\not\parallel$	$\supseteq$			$\not\subseteq$
$\not T$			$\not T$				$\not\supseteq$	

Table 1: The set relation transitivity table.

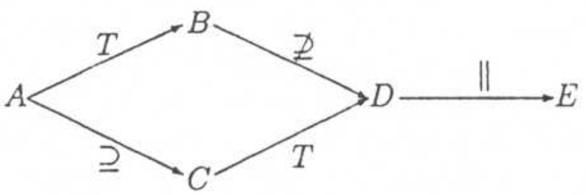

Figure 3: A network of relations. Graph search using the transitivity table reveals that $A \supset E$.

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 uses $A \not\parallel B$ in its transitivity search.

3.2 Deriving Relations through 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 contains x_1, x_2, and x_3 as known elements 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.

Table 2 presents the 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

Relationships between set A and set B

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

Table 2: Using membership comparison to derive ordinal relationships.

first set are members of the second *and* that the first set is closed, that is, the set has no members other than those explicitly enumerated.

The complexity of the membership comparison algorithm is $O(N \log N)$, where N is the number of known elements in the sets, while the path searching algorithm is $O(R)$, where R is the number of relations asserted between sets. In the problems we have encountered, the membership comparison mechanism is more efficient than the path search mechanism since N is typically much less than R. Hence, our strategy is to use membership comparison first and try path search only if more information remains to be derived.

4 Closure

Asserting that a set is closed means that the only members of the set are those currently known to the system. 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 each element currently a member of the set, so that it is retracted if any of the membership propositions are retracted.

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 3.2), relations such as subset or disjoint cannot be determined unless it is known that one of the sets or a relative 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, SERF infers that $x \in \bar{B}$.

A further use of closure is in detecting inconsistencies. In particular, it is inconsistent to assert that an element is a member of a closed set (unless, of course, it already is a member of the set). For example, asserting $A = \emptyset$ and $x \in A$ causes the constraint propagation mechanism to assert that $x \in \emptyset$ then flag an inconsistency because $\emptyset$ is empty and closed.[1]

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 — 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.

5 Cardinality

SERF provides mechanisms for describing and reasoning about the cardinality of sets. The mechanism uses the Quantity Lattice [6] to reason about inequalities, 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 ordinal relationship $(<, \leq, >, \geq, =, \neq)$ 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.

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 for George to be a parent of John would violate the cardinality constraints.

6 Summary

SERF is a utility for generic set reasoning that integrates mechanisms for propagat-

[1]When an inconsistency is detected, the TMS presents the user with the option of removing it by retracting one of its underlying assumptions.

ing membership propositions, deriving relations between sets, and reasoning about closure and cardinality. The central constraint network mechanism supports multiple modes of inference, such as local propagation and graph search. We have found a comprehensive set reasoner to be useful in several domains and expect these techniques to be applicable to the wide variety of commonsense reasoning tasks involving sets.

Acknowledgment

Yishai Feldman, Alex Yeh and Brian Williams contributed helpful comments on an earlier draft. This work was supported in part 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

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

[2] Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 12(2):231–272, 1979.

[3] David A. McAllester. *ONTIC: A Knowledge Representation System for Mathematics.* TR 979, MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA, 02139, 1987.

[4] Raymond Reiter. On closed world data bases. In H. Gallaire and H. Minker, editors, *Logic and Data Bases*, pages 55–76, Plenum Press, 1978.

[5] Reid Gordon Simmons. *Representing and Reasoning About Change in Geologic Interpretation.* TR 749, MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA, 02139, 1983.

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

[7] 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.

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

[9] Michael Paul Wellman. *Reasoning about preference models.* TR 340, MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA, 02139, May 1985.