TAXONOMIC REASONING

Josh D. Tenenberg
Dept. of Computer Science
University of Rochester
Rochester, NY 14627
(716)275-5671
josh@rochester

Abstract

In formalizing knowledge for common sense reasoning, one often needs to partition some domain. An instance of this from the Blocks World is the statement "All blocks are either held, on the table, or on another block." Although we can write this axiom in predicate calculus or in clause form for input to a theorem prover, such representations are highly space inefficient. In this paper we present a generalized clause form that allows for the compact representation of arbitrary partitions, along with a set of corresponding inference rules. Additionally, a theorem prover implementing these rules is described that demonstrates their utility with certain kinds of common sense rule bases.

1. Introduction

Reasoning about taxonomies has received attention, most notably within graph-structured representations. Shapiro (1979) introduces the AND-OR operator ( ), which when applied to a set of *n* literals indicates that between i and *j* of them, inclusive, are true. This has a rather simple representation within a semantic network, although the details of its use and properties are embedded within the graph search procedures. Hendrix (1979) develops a partitioned associative network through the judicious use of subset and disjoint arcs. Stickel (1982) discusses taxonomic reasoning by using a search through a connection graph built from a set of assertions. The work described in this paper contrasts with those cited above in combining its use of purely declarative knowledge, its generality in allowing one to reason about non-ISA as well as ISA partitions, and its simple inference rules, which can be added to an existing declarative theorem prover to enable more efficient taxonomic reasoning.

2. Representation

Two examples that are typical of commonsense reasoning are "All liquids either rest on a surface, rest in a container, flow along a surface, flow within a conduit, or fall in free space" (Hayes 1978) and "For any two times t_1 and t_2, either $t_1<t_2$, $t_1=t_2$, or $t_1>t_2$". These kinds of partitions are more general than typical ISA hierarchies. We can represent the latter statement using predicate calculus in conjunctive normal form (CNF) by

$$(t_1<t_2 \vee t_1=t_2 \vee t_1>t_2) \wedge [\sim(t_1<t_2) \vee \sim(t_1=t_2)] \wedge [\sim(t_1=t_2) \vee \sim(t_1>t_2)] \wedge [\sim(t_1<t_2) \vee \sim(t_1>t_2)],$$

where the first disjunction indicates that one of the conditions must hold, and the next three indicate that not more than one can hold. Unfortunately, as the number of partitions grows linearly, the CNF formula grows quadratically, as all partitions must be pairwise compared. We will introduce an n-ary exclusive-or operator, X, a syntactic variant of a connective used in (Hayes 1978) to capture this relationship among the partitions, and will write the above axiom

$$X(t_1<t_2,\ t_1=t_2,\ t_1>t_2).$$

The truth table for X is true just when exactly one of the literals inside is true, false otherwise. Using this notation, the expression of axioms like the above require a linear amount of space relative to the number of atoms, rather than the quadratic amount previously needed. This notation can be generalized, and any expression of the form

$$X(B_1,\ B_2,\ \ldots,\ B_n)$$

where the B_1 are literals will be called an *X-bundle.* The equivalent CNF formula, which will be called the Expand of $X(B_1, B_2, \ldots B_n)$ is the conjunction of all members of the set

$$\{B_1 \vee B_2 \vee \ldots \vee B_n\} \cup \{\sim B_i \vee \sim B_j \mid 1 \le i < j \le n\}.$$

The degree of an X-bundle is the number of literals within it, so that the above X-bundle has degree *n.* Note that an X-bundle of degree one is equivalent to the literal it contains, and will usually be identified with that literal.

An expression having the form

$$A_1 \vee A_2 \vee \ldots \vee A_n$$

where each of the A, are X-bundles, rather than simple literals, will be called an *X-clause.* This form reduces to regular clausal form when the degree of each A, is one. If only a single A_1, has degree greater than one, we will call this a *singular* X-clause. A conjunction of X-clauses is in exclusive normal form (XNF), and as usual, we will think of this as a set Any set of clauses in CNF is therefore trivially in XNF. Converting the X-clause $A_1 \vee A_2 \vee \ldots \vee A_n$ where each A, is an X-bundle, to a CNF clause produces

$$\{a_1 \vee a_2 \vee \ldots \vee a_n \mid a_i \in \text{Expand}(A_i)\}.$$

This work was supported in part by an Office of Naval Research Dept. of the Navy grant number N00014-80-C-0197 and the Rome Airforce Development Center

The space savings can thus theoretically be an exponential based on the number of X-bundles, although in practical cases using singular X-clauses (as defined above), the savings will be quadratic.

3. Inference Rules

A sound and complete set of inference rules for a set in XNF has been developed, and a presentation with proofs is given in (Tenenberg 1985). This set of rules was developed from looking at all of the ways in which an atom can occur twice in one parent clause, as in factoring, or in two parent clauses, as in binary resolution (Robinson 1965). Essentially, these inference rules involve unbundling the X-clauses slowly, peeling off the constituent X-bundles one at a time. Although all rules below are stated for propositions, the same rules apply to predicates, where the matching is between unifiable atoms.

The first rule generalizes binary resolution, and will be called X-resolution, which states that from the X-clauses

$$X(B, \underline{C}) \vee \alpha_1 \text{ and } X(\sim B, \underline{E}) \vee \alpha_2$$

we can infer

$$X(\underline{C}, \underline{E}) \vee \alpha_1 \vee \alpha_2.$$

(The *as* indicate the remaining disjunction of X-bundles in the clause, and the underlined capital letters indicate the remaining literals in the X-bundle).

With this rule

Block(A) and
Block(y) ⊃ X(OnTable(y), Held(y), OnOther(y)).

entail

X(OnTable(A), Held(A), OnOther(A)).

Additionally, using the same rule

X(Arch(B), Block(B), Wall(B)) and
X(~Arch(B), Support(B), Tower(B))

entail

X(Block(B), Wall(B), Support(B), Tower(B)).

Likewise X(Wall(y), Block(y), Arch(y)) and
Wall(z) ⊃ Stationary(z)

we can infer

~Stationary(y) ⊃ X(Block(y), Arch(y)).

The second rule, called exclusion, unifies atoms that are the same, rather than complementary, and stems from a very simple fact - if we know that exactly one of some set of conditions holds, and we also know that one of these conditions does in fact hold, then we can infer that all of the rest do not hold. For example, from the X-clauses

OnTable(A) and
X(OnTable<A), Held(A), OnOther(A))

we can infer both

~Held(A) and -OnOthertA).

The general form of this rule states that from the X-clause parents

$$X(B, \underline{C}_k) \vee \alpha_1 \quad \text{and} \quad X(B, \underline{E}_n) \vee \alpha_2$$

(where the subscripts on $\underline{C}$ and $\underline{E}$ indicate the number of remaining literals in their respective X-bundles) we can infer any member from the set of clauses

$$\{\sim E_i \vee X(\underline{C}_k) \vee \alpha_1 \vee \alpha_2 \mid 1 \le i \le n\} \cup$$
$$\{\sim C_j \vee X(\underline{E}_n) \vee \alpha_1 \vee \alpha_2 \mid 1 \le j \le k\}.$$

Using this rule in a more complex case, from the X-clauses,

Block(y) ⊃ X(Held(y), OnTable(y), OnOther(y)) and
X(Held(B), WeighsATon(B))

when we unify on Held, we can infer the following three X-clauses

Block(B) ∧ WeighsATon(B) ⊃
X(OnTable(B), OnOther(B)),
Block(B) ∧ OnTable(B) ⊃ WeighsATon(B),
and Block(B) ∧ OnOther(B) ⊃ WeighsATon(B).

To guarantee completeness, there are also four inference rules needed for the various ways in which atoms can co-occur within a single X-clause: (1) the same atom within the same X-bundle -- $X(A,A,\underline{C})$, (2) the same atom in different X-bundles -- $X(A,\underline{C}) \vee X(A,\underline{E})$, (3) complementary atoms in the same X-bundle -- $X(A,\sim A,\underline{C})$, and (4) complementary atoms in different X-bundles -- $X(A,\underline{C}) \vee X(\sim A,\underline{E})$. These rules happen to be the worst computationally, but they also cover the least likely cases. A restricted version with better computational properties will be listed later.

4. Restricted Inference Rules

A prototype theorem prover was built to implement these rules. It soon became clear that adding entire sets of clauses to the data base from a single inference is costly due to the fact that each of these new clauses must be checked as possible parents in future deductions, although few if any are likely to be needed for a proof. To deal with this and other practical problems that were encountered, several restrictions were added. When an X-rule results in more than one child, this indicates that some X-bundle is being decomposed, effectively negating the benefit of the representation. Thus, the inferences are ordered to defer such decomposition for as long as possible, doing so only when all children will be unit clauses. Take as an example the four clauses "~A", "B_i", "~C", and "$A \vee C \vee X(B_1, ..., B_n)$". If we first exclude with B_1, the set $\{A \vee C \vee \sim B_i \mid 2 \le i \le n\}$ is inferred, and ~A and ~C are now unifiable with each member. If we continue this way, there are, in the worst case, roughly 5n possible inferences that we can make within just this clause set. Had we written this clause set in CNF, it would have required on the order of $n^2/2$ clauses, with greater than n^2 possible inferences. If, however, we use the X-rules with the restriction mentioned above, only 3 inferences are made: the resolutions with ~A, then with ~C, and the exclusion on this descendant clause with B_1.

Another restriction is that only singular X-clauses are allowed in the database (clauses having at most one X-bundle of degree greater than one), and inferences are performed only if they produce singular X-clause children. A selection strategy that combines unit resolution and set of support is used. Additionally, two types of subsumption are used. The first eliminates clauses subsumed by a unit clause, while the second eliminates parents whose children subsume them, a common occurence when using unit resolution on fully instantiated clauses. The complete set of rules that are implemented are below, with informal proofs of their correctness.

The first three deal with intra-clausal matching of atoms (the remaining X-bundles were left off of clauses for notational clarity). The rule for matching complementary atoms in different X-bundles was entirely eliminated, since it always produces a linear number of non-unit children.

1) $X(A, \sim A, \underline{B}_k) \vdash \{\sim B_i \mid 1 \leq i \leq k\}$

Since one of A or ~A must be true, all of the B,'s must be false.

2) $X(A, A, \underline{B}) \vdash X(\underline{B})$ and $\sim A$

In this case, since only one of the literals within the bundle can be true, A must be false, and the true literal must be one of the B's.

3) $A \vee X(A, \underline{B}) \vdash A \vee X(\underline{B})$

If A is true, the first A establishes the truth of the clause independent of the following X-bundle. If A is false, then one of the B's must be true.

The next two rules are special cases of X-resolution, and the third is a restriction on exclusion.

4a) $X(\sim A, \underline{B}), A \vdash X(\underline{B})$

4b) $\sim A \vee \alpha, A \vdash \alpha$

5) $X(A, \underline{B}_k), A \vdash \{\sim B_i \mid 1 \leq i \leq k\}$

5. Implementation

This set of rules and strategy was quite powerful within several domains, although committing to unit resolution sacrificed completeness. Using only the set of general inference rules and strategy outlined above, with no domain specific rules, such as weighting of certain clauses or constants, there were several proofs that made only a few more inferences than the optimal proof tree. Figure 1 gives the proof tree that the theorem prover generated for a simple problem. All of the unit clauses were initially placed in the set of support, and the boldfaced clauses are subsumed by following clauses. Only one superfluous inference was made in a possible space of several hundred.

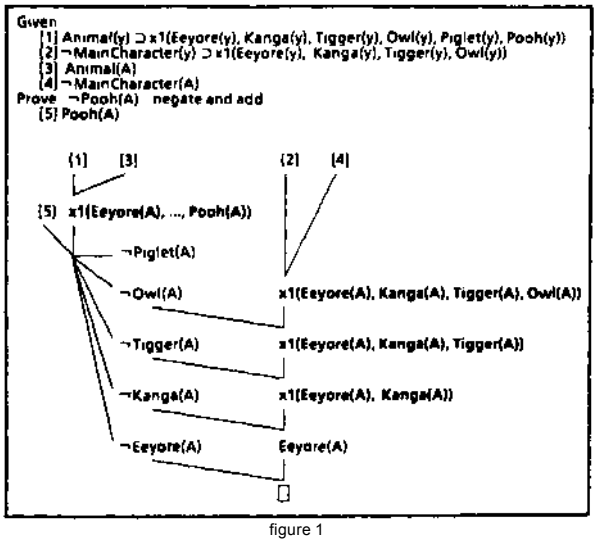

figure 1

Another example was run using a commonsense rule base with 50 X-clauses, that contained an axiomatization of a highly partitioned domain, with such axioms as

PlaysGolf(z) ⊃ X(Pensioner(z), KxMovieStar(/))

Building(z) ⊃ X(Fasti oodJoint(/), Factory(/), Jail(z)).

In typical proofs requiring 8-12 separate inferences, only 3 to 5 times this number of clauses was generated. Even writing such a database for a clausal theorem prover would require 115 clauses, with the same proofs requiring at least 10-15 separate inferences. This space savings combined with shorter proof depth is significant, when one considers that in general, search spaces for proofs grow exponentially, thus raising the hope that more proofs will be made tractable using this method.

6. Conclusion

A logical form with corresponding inference rules has been presented that is useful in representing and reasoning within partitioned domains. Not only are declarative data bases smaller to write using the methods presented here, but fewer deductions are typically needed for a proof than using standard resolution. Due to the simplicity of its clause form and inference rules and its similarity to resolution-type systems, this extension can be added to many existing theorem provers with a minimum of effort, enabling better reasoning capabilities within taxonomic domains.

Special thanks to Pat Hayes, whose wit, insights and criticism were invaluable, and Leo Hartman, who never seemed to lose curiosity nor tire of my long-winded explanations.

RFFFRFNCFS

[1] Hayes, P.J. "Naive Physics E: Ontology for Liquids", Working Paper 63, Institut pour les Etudes Semantiques et Cognitives. Geneva, 1978.

[2] Hendrix, G.C. "Encoding Knowledge in Partioned Networks" in Associative Networks, cd. Findler, N.V. 1979.

[3] Robinson, J.A. "A Machine-Oriented Logic Based on the Resolution Principle" Journal of ACM 12 (1965) 23-41.

[4] Shapiro, S.C. "The SNePS Semantic Network Processing System", in Associative Networks, ed. Findler, N.V. 1979.

[5] Suckcl, M.F. "A NonClausal Connection-Graph Resolution Theorem-Proving Program", Technical Note 268 SRI International. Menlo Park, CA Oct. 1982.

[6] Tenenberg, J.D. "Reasoning Using Exclusion: An Extension of Clausal Form", Technical Report 147 University of Rochester, Rochester. NY 1985.