Preserving Consistency across Abstraction Mappings

Josh D. Tenenberg

Computer Science Department
University of Rochester
Rochester NY 14627

Abstract

An abstraction mapping over clausal form theories in first-order predicate calculus is presented that involves the renaming of predicate symbols. This renaming is not 1-1, in the sense that several predicate symbols Ri,..., R_n from the original theory are all replaced by a single symbol R in the abstract theory. In order to preserve consistency, however, the clauses that distinguish the Rj's must be discarded in the abstract theory. This leads to a simple semantics; the union of the extensions of each of the Ri's in any model of the original theory forms the extension of R in a model of the abstract theory.

1 Introduction

An important method of constraining search in combinatorially intractable problems is to map the representation of the problem into an abstract representation, solve the problem in the abstract search space and use the abstract solution to guide the search in the original problem space. Work of this nature has been pursued by many researchers, most notably [Amarel 1972, Hobbs 1985, Korf 1985, Plaisted 1981, Sacerdoti 1974]. Of importance in this work is that certain formal properties hold between the original and the abstract versions of the problem in order to justify the use of abstraction. In this paper we will define a syntactic abstraction mapping between two first-order theories that involves the renaming of predicate symbols, a subclass of those mappings first defined in [Plaisted 1981]. The mapping is demonstrated to have the following properties. First, the abstract thoery will be consistent if the original theory is; second, for each abstract theorem T" there exists a theorem T in the original theory for which T is an abstraction; third, the model-theoretic semantics of the abstract theory is intuitively appealing.

In section 2 we present the formal definition of a *predicate mapping*, and show in section 3 both its relation to Plaisted's abstraction mappings, and its inherent problem of generating inconsistencies at the abstract level. Section 4 provides a set of syntactic restrictions that satisfies a particular semantics, and Section 5 demonstrates that these restrictions overcome the inconsistency problem by showing constructively that a model exists for the abstract theory given a model of the original theory. Section 6 will consist of a small example, and section 7 finishes with concluding remarks.

This work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Boiling AFB, DC 20332 under Contract Number F30602-86-C-0008 which supports the Northeast Artificial Intelligence Consortium (NAIC).

The author would like to thank the Xerox Corporation University Grants Program for providing equipment used in the preparation of this paper.

2 Predicate Mappings

We will assume standard first-order logic terminology as in [Enderton 1972]. In addition, all wffs will be taken to be in clause form [Robinson 1965]. Recall that a clause is a set of literals, and that clause C *subsumes* clause D if there exists a substitution ϕ such that Co is a subset of D. In addition, C and D are *variants* if C and D are instances of one another; that is, they are identical except for a renaming of variables.

Predicate mappings are functions that map predicate symbols from one first order language to those of another. Given two sets of predicates Pi, P2,

$f: P_1 - P_2$,

where the Pi are the only symbols that possibly distinguish the first order languages Li and L»2. What is noteworthy about f is that it is not necessarily 1-1, and therefore more than one relation symbol from I4 can map to the same relation symbol from L2. We can then define f over literals such that literals in Li are mapped to literals in L2 by replacing the predicate symbols under f. Likewise, f can be extended to clauses and sets of clauses in precisely this way.

3 Plaisted's Abstraction Mappings

Predicate mappings are a subclass *of abstraction mappings* that are defined in [Plaisted 1981]:

DEFINITION. An *abstraction* is an association of a set f(C) of clauses with each clause C such that f has the following properties:

(1) If clause C3 is a resolvent of CI and C2 and D3 e fl(C3), then there exist DI 6 fl(CI) and D2 * ftC2) such that some resolvent of DI and D2 subsumes D3.
(2) RNIL) = {NIL}. (NIL is the empty clause.)
(3) If CI subsumes C2, then for every abstraction D2 of C2 there is an abstraction DI of CI such that DI subsumes D2.

If f is a mapping with these properties, then we call f an *abstraction mapping* of clauses.

In addition, Plaisted proves the following theorem:

THEOREM. Suppose o is a mapping from literals to literals. Let us extend _ to a mapping from clauses to clauses by $\phi(C) = \{\phi(L) : L \in C\}$. Suppose o satisfies the following two properties:

(1) $\phi(\neg L) = \neg(\phi(L))$. That is, ϕ preserves complements.

(2) If C and D are clauses and D is an instance of C, then $\phi(D)$ is an intance of $\phi(C)$. That is, ϕ preserves instances.

Then ϕ is an abstraction mapping.

By this theorem, predicate mappings are abstraction mappings since they preserve both complement and instance.

Abstraction mappings have the property that from every clause C derivable from a set of clauses S, there exists a clause C derivable from AS) that subsumes an abstraction of C (proven in [Plaisted 1981]). Informally, we might say that every solution in the original problem space has a corresponding solution in the abstract space. This will be termed the *upward-solution property.*

A problem with abstraction mappings, however, (and hence with predicate mappings) is that they may result in abstract theories which are inconsistent even though the original theory is consistent, which Plaisted termed the "false proof' problem. As an example, suppose we have a clause set for a simple domain in which we have objects that are bottles ana glasses, with clauses stating that they are disjoint (note that all clauses will be in implication form for ease of interpretation):

1) $\mathbf{Bottle(x) \supset \neg Glass(x)}$,
2) $\mathbf{Glass(x) \supset \neg Bottle(x)}$,

and the clause stating that object A is a bottle:

3) Bottle(A).

If our predicate mapping f maps both Bottle and Glass to a new predicate *Container* in the abstract theory, then from the abstract theory we can derive a contradiction:

$\mathbf{Container(x) \supset \neg Container(x)}$
$\mathbf{Container(A)}$ $\vdash$ - Contained A)

The approach taken in the remainder of the paper is to restrict the original set of clauses over which the predicate mapping can be applied so as to preserve consistency.

4 Restricted Predicate Mappings

The intuition behind the semantics of *restricted* predicate mappings is that we would like the interpretation of a predicate in the abstract theory to be the union of the interpretations of each of the predicates in the original theory that map to it. So the objects that are Containers are all of those objects that are either Bottles or Glasses, or any of the other things that map to Container.

We can obtain this interpretation syntactically by removing all of those axioms from the original theory that serve to distinguish the relations that are conflated at the abstract level. Relations Ri,..., R_n in axiomatization 0^1, can be made indistinguishable in axiomatization 82 by systematically replacing each Ri with a new symbol R' under some predicate mapping f as before, but including in 62 only those clauses flC) such that C is in Bi and every clause mapping to C is derivable from θ_1. So, for instance, if we wish to conflate the relations Glass and Bottle by mapping them to the same symbol, say Container, then given the axioms

$\mathbf{Glass(x) \supset Breakable(x)}$
$\mathbf{Bottle(x) \supset Breakable(x)}$
$\mathbf{Bottle(x) \supset Corkable(x)}$

we would have in the new clause set only

$\mathbf{Container(x) \supset Breakable(x)}$,

since Corkability is true only of Bottles, and therefore distinguishes Bottles from Glasses in the original theory, while Breakability is true of both Bottles and Glasses. In other words, the clause

$\mathbf{Container(x) \supset Corkable(x)}$

would not be placed in the new clause set since one of the clauses that maps to it, namely

$\mathbf{Glass(x) \supset Corkable(x)}$

is not derivable from the original clause set.

This constraint, however, is stronger than required. For instance, it keeps one from ever asserting in the abstract theory that an object is a container in theories in which bottles and glasses are disjoint, as in (1) - (3) above, since it will never be the case that an object in the original theory is both a bottle and a glass. Given the desired semantics, it is permissible to allow the inclusion of mapped *positive* clauses from the original theory, those clauses containing no negated literals, into the abstract theory. That is, if obiect A has property P, then certainly object A will have the abstract property f(P), and likewise for predicates of any arity. Thus, one can map the clause Bottle(A) to ContainerA), since all bottles are containers, even though Glass(A) may not hold. In addition, this allows the mapping of clauses such as Bottle(A) v Glass(A) to Container(A) v Container(A).

5 Formal Definitions and Proof of Consistency

We use [C]f to denote the set of symbols or clauses that map into C under predicate mapping f. That is,

$[C]_f = \{D \mid f(D) = C\}$.

Each such D will be called a *specialization* of C. If 9 is a clause set and D is a clause then we will use $\theta \vdash D$ to mean that the null clause can be derived from the clause set $\theta \cup \neg D$ by the inference rule *full resolution* [Robinson 1965]. Let 9 be an axiomatization in clause form of a theory over language L_1 and let f be a predicate mapping from L_1 to L_2. We define $g(\theta)$, a *restricted predicate mapping,* to be a subset of $f(\theta)$, where

(4) $g(\theta) = \{C \mid$ there exists some $D \in [C]_f$ such that $D \in \theta$ and either C is a positive clause or for every $D \in [C]_f$ it is the case that $\theta \vdash D\}$.

The main theorem states that given a clause form axiomatization 6 and a restricted predicate mapping g, g(6) is consistent if θ is. In other words, consistency is preserved by the mapping. Given that a first-order clause set 9 is consistent if and only if it is satisfiable, i.e., there exists a model that assigns truth to all of the clauses D such that $\theta \vdash D$, this theorem will be proven by constructing a model of g(9) from a model of θ.

Standard formal semantic terminology will be assumed [Enderton 1972]. In particular, an *interpretation* is a

structure <D, G>, where D is a set of objects, the *domain,* and G is an *interpretation function* that assigns to each constant symbol an object from D, to each n-function symbol an n-function mapping objects in D to an object in D, and to each n-predicate symbol an n-relation over objects in D. Furtner, we say that y is a *value assignment* if it assigns to every variable symbol an object in D. If o is a symbol from the language over which G is defined, we say that G(o) is the *interpretation* of o under M, which in the case of an n-predicate will be a set of n-tuples. In addition, clauses are assigned *truth values* according to semantic rules pertaining to the logical symbols, which in the case ofclause form is universal quantification, negation, and disjunction. If o is a clause and y is a value assignment, then M(o) returns either True or False under y, denoted by My(o). IfMy(a) =Truefor every value assignment y, then we say that o is True in M, or M(o)=True, otherwise o is False in M. An interpretation M = $\mathbf{<D, G>}$ is a *model* of a clause set θ if D is True in M for every clause D such that $\theta \vdash \mathbf{D}$.

Theorem:

Let 8 be an axiomatization in clause form of a theory over language L_1, let f be a predicate mapping from L_1 to L_2, and let g be the restricted predicate mapping defined relative to fas in (4) above. For any model M= $\mathbf{<D, G>}$ of θ, $\mathbf{M' = <D, G'>}$ is a model of $\mathbf{g(\theta)}$ where for every pi[illegible] of L_2,

$$\mathbf{G'(\sigma) = \bigcup_{\xi \in [\sigma]_f} G(\xi)} \quad (5)$$

and for every non-predicate symbol o, $\mathbf{G'(\sigma) = G(\sigma)}$.

Proof:

Let us assume that M' is not a model of g(0). There must then be some clause $\mathbf{C \in g(\theta)}$ that is False in M' There are two cases to consider - first where C is a positive clause, and second where C contains negated erals.

Case 1: Assume C is a positive clause.

For membership of C in $\mathbf{g(\theta)}$, it must be that there exists some clause E € [C]f such that $\mathbf{E \in \theta}$, by the definition of g. Since M is a model of 8, E is True in M. Then, for every value assignement y, My(E)=True. Let us take truth in M andM' relative to a particular value assignment y. E is of the form $\mathbf{E = \{P_1(a_1), ..., P_n(a_n)\}}$, where ai represents the predicate Pi's list of arguments. My(E)=True means that My(Pi(ai)) = True for some literal of E. Since ai represents the terms (an,..., ajj) of Pi, then $\mathbf{<G(a_{i1}), ..., G(a_{ij})> \in G(P_i)}$. Additionally it follows that $\mathbf{<G'(a_{i1}), ..., G'(a_{ij})> \in}$ G'(flPi)) under y, by the definitions of the interpretation function G given in (5) above, and the predicate mapping f. That is, for $\mathbf{Q = f(P_i)}$, $\mathbf{M'_Y(Q(a_{i1}, ..., a_{ij})) =}$ True, but this literal is in C, and so M'y(C)=true. Therefore, $\mathbf{M'_Y(C)}$ = True for every value assignment y since y was chosen arbitrarily, making M'(C)=True, contradicting our assumption. This dispatches the first case.

Case 2: Assume C contains negated literals.

For membership of C in g(8), for every clause $\mathbf{E \in [C]_f}$, 9 h E and some $\mathbf{E \in \theta}$, l the definition of g. Since M is a model of θ, $\mathbf{M(E)}$ = True for every such E, and therefore My(E)=True for every value assignment y.

Since C is False in M\ there must exist a value assignment y such that M'y(C) = False. Let us take truth in M and *M* relative to this value assignment y. C is of the form $\mathbf{C = \{R_1(a_1), ..., R_n(a_n), \neg S_1(b_1),}$ ..., -» Sk(bk)K where ai and bi are the arguments of each of the literals. That is, some of the literals in C are negated, and some may be positive. For every literal L *i* C, MY) = False. This means that for every E *i* [C]f, for every positive literal W *z* E, My(W) = False, since were it not, then M y(g(W))=True for some literal W and hence M'y(C)=True by the definition of G', contradicting our assumption that $\mathbf{M'_Y(C) = False.}$ Therefore, under this value assignment one or more of the negated literals in each E e [C]f must be True.

Since each -Si(bi) is False in NT under y, then each Si(bi) is True. That is, taking bi to represent the ground terms (bn,..., by) of each $\mathbf{\neg S_i}$, $\mathbf{<G'(b_{i1}), ..., G'(b_{ij})> \in G'}$(Si) for each of the negated literals under y. But by the definition of G', for each of the Si's there exists some $\mathbf{Q_i \in}$ [Si]f where $\mathbf{<G(b_{i1}), ..., G(b_{ij})> \in G(Q_i)}$ under this same value assignment. That is, My($\mathbf{\neg Q_i(b_i)) =}$ False for some specialization Qi of each predicate Si of each negated literal in C. Let E be the clause consisting of the negation of each such Qi(bi) and some specialization of each of the positive literals in C. Note tnat $\mathbf{E \in [C]_f}$. My(E) = False, since each of the literals in E are False. This then contradicts the earlier conclusion that every E *i* [C]f be assigned True by M for every value assignment y. This dispatches case 2, and hence establishes tne theorem, QED

Restricted predicate mappings are no longer abstraction mappings by Plaisted's definition, since the upward-solution property is notpreserved. However, abandoning the completeness afforded by the upward-solution property is likely what is required to solve frequently encountered problems at a lower cost than if they were solved directly in the original theory. These restricted mappings do have what can be termed the *downward-solution property,* since a trivial corollary of the theorem is that *for* every clause derivable from tne abstract theory, there will be a specialization of it derivable from the original theory. Note that there may be no solution to tne problem in the abstract theory since its solution requires some of the details that are ignored at that level. However, every abstract solution is guaranteed to have a specialized solution to the original problem.

An important objection that one might have is that the syntactic mapping function *g* given in (4) is undecidable, since it is based upon determining 8 - D for every clause D mapping to each candidate clause in the abstract clause set. One should note, however, that the search for derivability can always be arbitrarily bounded, and if no proof is obtained within this bound then it can be assumed that this clause is *not* derivable. In this way, consistency is still preserved between the original and the abstract theory, the abstract theory being simply weaker than it theoretically could be, i.e., having fewer theorems. The appropriate amount of resource that one should expend in constructing abstract theories is quite an important issue but outside the bounds of this paper. These kinds of performance choices, however, are discussed in [Hartman and Tenenberg 1987], elsewhere in this volume.

6 Example

Let θ, the original clause set, be the following clauses (note that A ⊃ B is equivalent to ¬A ∨ B, and is therefore not a positive clause):

1) Bottle(x) ⊃ MadeOfGlass(x)
2) Bottle(x) ⊃ Graspable(x)
3) Glass(x) ⊃ MadeOfGlass(x)
4) Glass(x) ⊃ Graspable(x)
5) Glass(x) ⊃ Open(x)
6) Box(x) ⊃ Graspable(x)
7) Bottle(x) ⊃ ¬Glass(x)
8) Glass(x) ⊃ ¬Bottle(x)
9) Bottle(x) ⊃ ¬Box(x)
10) Glass(x) ⊃ ¬Box(x)
11) Box(x) ⊃ ¬Glass(x)
12) Box(x) ⊃ ¬Bottle(x)
13) Bottle(x) ⊃ MilkBottle(x) ∨ Winebottle(x)
14) Open(x) ∧ Graspable(x) ⊃ Pourable(x)
15) Graspable(x) ⊃ Movable(x)
16) MadeOfGlass(x) ⊃ Breakable(x)
17) Bottle(x) ∨ Glass(x) ∨ Box(x)
18) Open(A)

Let f_1 map each predicate symbol to itself except that it maps *Bottle* and *Glass* both to *GlassContainer*. Therefore, $g_1(\theta)$ is the following:

1') GlassContainer(x) ⊃ MadeOfGlass(x)
2') GlassContainer(x) ⊃ Graspable(x)
6') Box(x) ⊃ Graspable(x)
9') GlassContainer(x) ⊃ ¬Box(x)
11') Box(x) ⊃ ¬GlassContainer(x)
14') Open(x) ∧ Graspable(x) ⊃ Pourable(x)
15') Graspable(x) ⊃ Movable(x)
16') MadeOfGlass(x) ⊃ Breakable(x)
17') GlassContainer(x) ∨ Box(x)
18') Open(A)

Note that clauses 3 and 4 are redundant in the abstract theory and are therefore not included in $g_1(\theta)$, as are 10 and 12. Clause 5 gets eliminated because *Bottle(x) ⊃ Open(x)* is not derivable from θ. Likewise with 7 and 8 as seen earler. Additionally, 17' was *factored* so that the same literal did not appear twice. Although $g_1(\theta)$ is much smaller than θ, there are still several intuitive inferences that can be made, such as *Graspable(x), Movable(x) and Pourable(A).*

Given the way in which abstraction is defined, $g_1(\theta)$ can additionally be treated as a primitive level theory and mapped into a more abstract theory $g_2(g_1(\theta))$. For instance, let f_2 map all predicates to themselves except *Box* and *GlassContainer* get mapped to *Container*. Thus, $g_2(g_1(\theta))$ is:

2") Container(x) ⊃ Graspable(x)
14") Open(x) ∧ Graspable(x) ⊃ Pourable(x)
15") Graspable(x) ⊃ Movable(x)
16") MadeOfGlass(x) ⊃ Breakable(x)
17") Container(x)
18") Open(A)

One can therefore abstract a theory arbitrarily many times, preserving consistency throughout. Each abstract theory has the advantage that those clauses that it can derive will likely be done so at considerably less cost than in the original theory. Although this would be true of any subset of θ, it is important to recognize that the mapping functions embed a significant amount of heuristic information about the domain. This function determines which clauses are important, and which are mere details, and so the choice about which clauses remain in the abstract theory is not done so on an arbitrary basis.

7 Conclusion

A useful way of viewing abstraction is as a function that maps one first-order theory into another. We have defined *predicate mappings* that involve the renaming of predicate symbols, where more than one of the original symbols can map into the same abstract symbol. These predicate mappings are a subclass of Plaisted's abstraction mappings, and as such suffer from the fact that consistency in the original theory may be lost in the abstract theory. To remedy this, we introduced syntactic restrictions to the predicate mapping that guarantee the preservation of consistency. These restrictions amount to eliminating the axioms in the original theory that distinguish the predicates which are being conflated in the abstract theory. This has a simple semantics; the extension of each of the predicates P in a model of the abstract theory can be constructed from the union of the extensions of each of the predicates that map to P in any model of the original theory. A proof of this was provided, along with an example demonstrating one possible use of the restricted predicate mappings.

Acknowledgements

I would like to thank my advisor, Dana Ballard, for his encouragement and wealth of ideas on abstraction. In addition, I would like to thank members of the UR-AI group, especially Leo Hartman, who always seems a step ahead, sometimes two, and Jay Weber, for pushing for a semantic definition. I am also grateful to Mark Fulk, for finding the problems with the *old* versions of the main proof, and pointing the way to the correct version.

References

Amarel, S., "On representations of problems of reasoning about actions", In D. Michie (editor), *MachineIntelligence* 3,131-171. American Elsevier, New York, 1972.

Enderton, H., *A Mathematical Introduction to Logic,* Academic Press, Inc., London, 1972.

Hartman, L., and Tenenberg, J., "Performance in Practical Problem Solving", Proceedings - IJCAI, Milan, Italy, 1987.

Hobbs, J,, "Granularity.", Proceedings - UCAI, Los Angeles, CA 1985.

Korf, R., "An analysis of abstraction in problem solving", Proceedings - ACM Technical Symposium on Intelligent Systems, 1985.

Plaisted, D., Theorem Proving with Abstraction", Artificial Intelligence 16:47-108, 1981.

Robinson, J., "A machine-oriented lone based on the resolution principle", Journal of the ACM 12:23-41, 1965.

Sacerdoti, E., "Planning in a hierarchy of abstraction spaces", Artificial Intelligence 5:115 -135, 1974.