Shifting Vocabulary Bias in Speedup Learning

DEVIKA SUBRAMANIAN

devika@cs.cornell.edu

Computer Science Department, Cornell University, Ithaca, NY 14853

Editor: M. desJardins

Abstract. In this paper, we describe a domain-independent principle for justified shifts of vocabulary bias in speedup learning. This principle advocates the minimization of wasted computational effort. It explains as well as generates a special class of granularity shifts. We describe its automation for definite as well as stratified Horn theories, and present an implementation for a general class of reachability computations.

Keywords: vocabulary bias, speedup learning, representation shifts, abstractions

1. Introduction

In recent years, researchers in machine learning have focused on the key problem of bias both in inductive and in deductive systems. Systems that learn concepts from labeled training instances employ inductive bias. Inductive bias is defined by Mitchell in (Mitchell, 1990) as “any basis for choosing one generalization over another, other than strict consistency with the observed training instances.” In the same paper, written in 1980, he concludes by stating that:

If biases and initial knowledge are at the heart of the ability to generalize beyond observed data, then efforts to study machine learning must focus on the combined use of prior knowledge, biases and observation in guiding the learning process. It would be wise to make the biases and their use in controlling learning just as explicit as past research has made their observations and use.

This paper is inspired by Mitchell’s observation about the importance of bias in designing learning methods, and the need for its declarative explication and use. While there has been a considerable amount of work on characterizing the nature of inductive bias in concept learning ((Utgoff, 1986), (Haussler, 1990), (Buntine, 1990), (Russell & Grosz, 1987), (Provost, 1992), (Gordon, 1990), (Spears & Gordon, 1992)), biases in speedup learning have received little attention.

Speedup learning algorithms improve performance of solvers over problem distributions (Rosenbloom, 1983). Learning problem distributions is one of the inductive components of speedup learning. The objective of a speedup learner is to find good tradeoffs between coverage of learned solvers over a problem distribution, and their time and space cost. Bias plays a key role in the design of speedup learners. Building on Mitchell’s, Utgoff’s, and Gordon and Spears’s definitions of inductive bias, we define bias for speedup learning as “any basis for choosing one solver over another for a given problem distribution”. We do not limit ourselves to solvers that only speed up observed training instances, thus

treating instances as additional sources of bias as in (Spears & Gordon, 1992). This paper makes two contributions to the understanding of bias shifts in speedup learning. First, it defines the types and sources of biases in speedup learning. Second, it develops and implements a domain-independent principle, called the *irrelevance principle*, for shifting vocabulary bias.

We consider a family of speedup learning methods that improve problem solving in theories formulated as function-free, definite Horn clauses. SLD resolution (Lloyd, 1987) is a complete inference method for such theories. Our results also apply to function-free, stratified Horn clauses ("safe" Horn rules with negated literals in the rule antecedents, for which SLD resolution with negation-as-failure is a complete inference procedure) with SLDNF resolution. We specify a solver as a pair (T, C) , where T is a definite or safe, stratified Horn clause theory and C is a control strategy (specification of rule ordering and conjunct ordering) for the interpretation of T by an inference engine employing SLD or SLDNF resolution. We can distinguish between two types of solvers, called one- and two-level solvers, based on whether the control strategy C is procedurally or declaratively specified. C is declaratively specified in a two-level solver and procedurally encoded in a one-level solver. Improvements in a one-level solver are therefore made by reformulating the theory T to which one has declarative access. Improvements in two-level solvers can involve reformulations of the base theory T , as well as the meta-level interpretation policy C . The class of one-level solvers encompasses classical formulations of theorem proving, planning and scheduling.

The major sources of bias in speedup learners are

1. Representational: how is the space of new solvers chosen? The constraints on coverage, average time and space cost of the learned solver impose few restrictions on its construction. The designer of a speedup learning algorithm fixes a set of operators that defines the space from which new (T, C) pairs are drawn. Operators that change the objects, functions and relations assumed by a (T, C) pair are said to change its *vocabulary bias*.
2. Algorithmic: how is the space of new solvers searched? How are solvers evaluated? Solvers are typically evaluated on the basis of coverage over a distribution, and average time and space costs. An algorithmic bias can be specified with a policy for trading off these evaluation dimensions against each other.

While the sources of bias in speedup learning are very similar at a high level to those in inductive learning, the details are quite different. The space of possible new solvers consistent with a training instance is difficult to calculate. The cost of evaluating a solver against a distribution and the cost of assimilating new instances are generally very high. There is, however, much more background knowledge available in the form of properties of the problem solver and the problem class, which can be put to use in designing initial biases and bias shifts. In contrast to inductive learning, it appears that a domain and encoding independent basis exists for choosing and shifting vocabulary bias for speedup learners. The goal of this paper is to declaratively describe and implement such a principle for vocabulary shifts in speedup learners.

We present the thesis that vocabulary bias shifts can be determined by a principle of computational economy (Subramanian & Genesereth, 1987) called the irrelevance principle. Explicit minimization of wasted computational effort can both justify and generate an interesting class of vocabulary shifts. The shifts sanctioned by the irrelevance principle are characterized by incremental coarsening of distinctions in a problem formulation. These transformations, which are information-losing in nature, make tradeoffs between coverage/accuracy and computational effort. They are routine in goal-directed spatial and temporal reasoning in humans. For instance, we conceptualize a road as a line when planning a trip, as a surface when we cross it, and as a volume when we attempt to dig it¹, and we move easily between point- and interval-based representations of time. We show the following in this paper.

1. We identify a parametric class of vocabulary bias shifts that simplify (T, C) pairs. We generate new vocabularies that make fewer distinctions than the original T . These vocabularies form a pre-order under the $\succeq$ (coarser-than-or-equal-to) relation.
2. We describe the irrelevance principle and develop a scheme for explicit reasoning about irrelevant computation. We show how it can be used to generate coarser theories with better computational properties.
3. We present an algorithm for incremental generation of shifts of vocabulary bias for definite Horn theories. We provide experimental results in the domain of reachability computations.

The paper is organized as follows. In Section 2, we provide a motivating example from the domain of reachability computations. In Section 3, we describe shifts of vocabulary bias, and introduce the class of granularity-coarsening transformations. We then propose the minimization of irrelevant computation as a basis for explaining and generating granularity-coarsening shifts. In Section 4, we present algorithms for the detection of irrelevant computation in definite as well as stratified Horn theories and schemes for repairing theories to eliminate wasted computation. We present experiments to validate our theory in the Gridworld domain (Tambe & Rosenbloom, 1989). We conclude in Section 5 with an evaluation of our approach and propose directions for future research in automating vocabulary shifts using domain-independent principles.

2. A motivating example

We begin our analysis with an example of a shift of vocabulary bias in the context of a function-free, definite Horn theory F_1 interpreted by a procedurally encoded control strategy C_o — SLD resolution, with rules chosen in order of entry, and conjuncts solved from left to right. SLD resolution is a sound and complete inference procedure for function-free, definite Horn theories on definite² goals.

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

We use the convention of denoting variables by capital letters and constants by small letters. `parent` is a binary relation on people in the universe of discourse. The literal `parent(X,Y)` stands for the fact that `x` is the parent of `y`. `parent` is non-reflexive, non-symmetric and non-transitive. The `parent` relation is representable as a forest of rooted trees. `ancestor` is the transitive closure of `parent`. `parent` is a primitive relation, and `ancestor` is a computed relation in (F_1, C_o) . Primitive predicates are solved by table lookup, while computed relations require inference for solution. The goal class D of interest is `samefamily( $X^b, Y^b$ )`, where both arguments are bound at the start of the computation. X^b means that variable X starts out being bound, while Y^f stands for the fact that Y starts out unbound or free. The goal predicate has a disjunctive definition. The last rule below says that two people belong to the same family if they have some common ancestor.

```
samefamily(X,X).
samefamily(X,Y) :- ancestor(X,Y).
samefamily(X,Y) :- ancestor(Y,X).
samefamily(X,Y) :- ancestor(Z,X), ancestor(Z,Y).
```

For the goal distribution D , the pair (F_1, C_o) solves every goal, as long there are no infinite descending chains in `parent`. We can establish that the average cost of solution is $O(k^c)$ where k is the average number of children per parent, and c is the average length of chains in `parent` explored during the solution of problems in D . A semi-automated derivation of this fact is presented in Appendix I. The space cost of (F_1, C_o) is dominated by the size P of the `parent` relation. If N is the number of people, then the size of the `parent` relation is $O(N)$.

The solver (F_2, C_o) preserves F_1 's coverage over D , has the same space cost as F_1 , but a time cost of $O(1)$. F_2 is shown below.

```
samefamily(X,X).
samefamily(X,Y) :- max_ancestor(X,Y).
samefamily(X,Y) :- max_ancestor(Y,X).
samefamily(X,Y) :- max_ancestor(Z,X), max_ancestor(Z,Y).
```

The `parent` relation is replaced by the `max_ancestor` relation where every person is associated with their maximal ancestor. The relation `max_ancestor` is definable in terms of the `ancestor` and `parent` relation. That is, there exists a first order formula that defines `max_ancestor` in F_1 .

$$\text{max_ancestor}(X,Y) \equiv \text{ancestor}(X,Y) \wedge \neg \text{parent}(Z,X)$$

(F_2, C_o) provides the same solution for every goal in D as (F_1, C_o) , since the identity of the common ancestor is not reported. The coverages of (F_1, C_o) and (F_2, C_o) are identical. The average cost of solution for (F_2, C_o) over D is $O(1)$, since each goal is solved by one or two lookups.³ The space cost of (F_2, C_o) is $O(N)$, where N is the number of people.

We can construct another solver with the same coverage and time cost as (F_2, C_o) but with a higher space cost. The solver (F_3, C_o) over the goal distribution D consists of a

table of `samefamily` pairs computed offline from (F_1, C_o) . The coverage of (F_3, C_o) is the same as that of (F_2, C_o) . The space cost is $O(N^2)$, since every pair of people in the same tree in the `parent` relation will be represented in the `samefamily` table. The time cost is $O(1)$ because it involves a single lookup in a table.

The vocabulary commitments of the theories F_1 , F_2 and F_3 are shown in the table below. $\mathcal{P} = \text{Domain}(\text{parent}) \cup \text{Range}(\text{parent})$ stands for the set of people in the database.

Theory	Objects	Primitive Relations	Computed Relations
F_1	$\mathcal{P}$	<code>parent</code>	<code>ancestor, samefamily</code>
F_2	$\mathcal{P}$	<code>max_ancestor</code>	<code>samefamily</code>
F_3	$\mathcal{P}$	<code>samefamily</code>	

We now pose the two questions raised in the introduction: how is the space of solvers to be generated and how is it to be searched? We provide a short preview of the answers in the context of this example. For the goal distribution D considered, the pair (F_2, C_o) is the “best” solver if complete coverage over the distribution is desired. F_2 can be automatically generated from F_1 by observing that from the point of view of D : (1) the distinction between immediate and non-immediate ancestry is irrelevant, and that (2) all intermediate ancestors are irrelevant, since the identity of the common ancestor is unneeded. These two facts about the irrelevance of distinctions in the original formulation with respect to the goal class provide justification for changing the relational ontology of the formulation. In the next two sections, we make this informal reasoning precise so that we can automatically generate vocabulary shifts for improving computational efficiency.

3. Evaluation of solvers and vocabulary bias

We first define the class of formulations considered in this paper. For expository ease, we describe our methods in the context of function-free definite Horn theories. All rules in a definite Horn theory are of the form $A \leftarrow B_1, \dots, B_m$ where the head A is a positive literal and the B_i ’s are also positive literals. Our results also hold for Horn theories with negated literals in bodies of the rules, as long as the rules are safe and stratified (Ullman 1989). A set of Horn rules is stratified if whenever there is a rule with head predicate p and a negated subgoal in the body with predicate q , there is no path in the dependency graph of the theory from p to q . A dependency graph of a Horn theory has predicate names as nodes; and arcs from p to q if there is a rule with p at its head and q (in positive or negative form) in its body. A rule is safe if the range of each of its variables is restricted either by the variable being an argument of a positive literal, or by it being equated to a constant or to another restricted variable through a chain of equalities. Negated literals cannot be used to restrict the range of a variable. SLDNF resolution (SLD resolution with negation by failure) is sound and complete for safe, stratified Horn theories with respect to definite goals. Every result in the paper holds

with the word *definite* being replaced by *safe and stratified* and *SLD* replaced by *SLDNF* resolution.⁴

3.1. Evaluating solvers

The goal of a speedup learner is to find the “best” solver $((T, C)$ pair) for a given goal distribution D . We evaluate solvers along three dimensions. Each dimension has a preference order defined on it.

1. Coverage E_C : it is the probability that (T, C) can solve a member of D . It is a real number in $[0, 1]$. For finite distributions, it can be expressed as the fraction of problems in D that (T, C) can solve. Coverage c_1 is better than c_2 , denoted $c_1 \succ c_2$, iff $c_1 > c_2$.
2. Average time cost E_T : it is the average time for (T, C) to solve a goal in D . It can be expressed in big- O terms with the calculus of big- O 's for comparisons. It can also be expressed as a real number with the standard $<$ ordering. Time cost t_1 is better than t_2 , denoted $t_1 \succ t_2$, if $t_1 < t_2$ when t_1, t_2 are reals.
3. Average space cost E_S : it is the size of the encoding of (T, C) . It can be expressed in big- O terms or as a real number. Space cost s_1 is better than s_2 , denoted $s_1 \succ s_2$, if $s_1 < s_2$, where s_1, s_2 are reals.

If the space and time costs are specified in big- O terms, they need to be converted to reals by user-specified functions which reflect the desirability of the various big- O classes. Given evaluation vectors $v_1 = (c_1, t_1, s_1)$ for a solver (T_1, C_1) and $v_2 = (c_2, t_2, s_2)$ for (T_2, C_2) , we say that v_1 *dominates* v_2 with respect to the distribution D , denoted $v_1 >_D v_2$, if $c_1 \geq c_2$ and $t_1 \leq t_2$ and $s_1 \leq s_2$ and at least one of the inequalities is strict. Solver (T_1, C_1) dominates solver (T_2, C_2) if $v_1 >_D v_2$. We use statistical sampling of the goal distribution to obtain estimates of coverage and average time cost to within user-specified confidence and error bounds (Subramanian & Hunter, 1992).

The coverage, space and time costs of the solvers (F_1, C_o) , (F_2, C_o) and (F_3, C_o) are plotted in Figure 1. We observe that (F_1, C_o) is dominated by (F_2, C_o) and (F_3, C_o) , and that (F_2, C_o) dominates (F_3, C_o) . If we wish to explore formulations with lower space cost than (F_2, C_o) , we have to eliminate some of the `max_ancestor` links and thus sacrifice on coverage. To compare solvers of this type with (F_2, C_o) , we need a value function $v : F \rightarrow \mathcal{R}$ that allows comparison of evaluation vectors that do not dominate each other. The value function is a mechanism by which the user can specify how he wishes to tradeoff coverage for time and space cost. Examples of such value functions include a linear combination, shown below, of the three parameters, with $k_i > 0$.

$$v(F) = k_1 E_C(F) + k_2 E_T(F) + k_3 E_S(F)$$

There are many other possibilities for defining value functions, including non-linear combinations of these three parameters. We can also consider refinements of these parameters. For instance, we can represent coverage as two numbers: one representing

the probability that the solver will provide an answer and one representing the probability that it will provide an incorrect answer. We can then weight these two types of coverage lapses differently. The algorithms in this paper are not sensitive to the specific value function chosen. They take the value function specified by the user as input and find local minima in the formulation space with respect to that value function.

Figure 1. Plotting the solvers in the kinship problem on the three-dimensional evaluation space. The projections on the time and space cost axes are sufficient to compare these solvers, since they all have the same coverage. Preferred directions are toward the origin for time and space cost and toward the point 1.0 for coverage. (F_2, C_o) dominates both (F_1, C_o) and (F_3, C_o) .

The goal of a speedup learner is to traverse the space of solvers starting with the initial solver, looking for a global optimum in the three-dimensional value space. An important issue is the recognizability of global optima. For instance, is (F_2, C_o) the globally optimal solver for the distribution D of $\text{samefamily}(X^b, Y^b)$ goals? For the model of space and time costs assumed, the only way we can improve upon (F_2, C_o) is to reduce its space cost to $O(1)$. This will result in a drop in coverage over D since max_ancestor links will be eliminated to save on space. Therefore, if the gain in the value function due to decreased space cost $k_3(E_S(F) - E_S(F_2))$ exceeds the value loss due to lost coverage $k_1(E_C(F) - E_C(F_2))$, we would prefer (F, C_o) over (F_2, C_o) for D . A point at which the two marginal costs become equal lies on an indifference curve in the coverage/space cost plane of the three-dimensional value space. All formulations on this curve have the same value for v , giving us a set of global optima. In this problem, we used specialized knowledge of the structure of the computation to determine how coverage and space are related. In general, however, there is no efficient method for verification of the global optimum short of examining the entire space. Thus, we have to be satisfied with local optima. To recognize a local optimum, we need to show that no one-step movement improves v . We therefore need to describe how the learner systematically generates such one-step movements.

3.2. Generating the space of vocabulary shifts

Each point (T, C) in the space of solvers embodies a specific ontological commitment, reflected in the set of objects, functions and relations in T and C . The change in ontological basis is implemented by a vocabulary shift. In this paper, we only consider changes in T 's ontology while preserving that of C . Such a study is a necessary precursor to a more general framework where both T and C can be varied. An example of a shift in C would be a granularity refinement over the C_o considered here to include a new predicate for literal ordering choice.

The *vocabulary* of T , denoted $Vocab(T) = (c, r, l)$, consists of the set c of named object constants in T , set r of relation symbols with their associated arities, and a map $l : r \rightarrow \{computed, primitive, hybrid\}$ that identifies the type of the relation symbols.⁵ We let $L(V)$ stand for the first-order language defined inductively on the vocabulary V . The theory T is a subset of the formulas in $L(Vocab(T))$. A general vocabulary shift is a mapping between the vocabularies of two theories.

There are three classes of vocabulary shifts: granularity-coarsening, granularity-refining and granularity-preserving shifts. In a granularity-coarsening shift, the new vocabulary can be constructed from the old vocabulary, but not vice versa. Thus distinctions are lost in the vocabulary shift. The shift to (F_2, C_o) from (F_1, C_o) is an example. This class of shifts encompasses all abstraction formations. Granularity-refining shifts introduce distinctions that are not present in the original formulation. So the new vocabulary cannot be constructed directly from the old one. An example of such a shift in the domain of Euclidean geometry proofs is the introduction of a perpendicular bisector to prove that the base angles of an isosceles triangle are equal. The space of possible refinement shifts is not easy to conceptualize – in fact, such shifts are likely to be problem-specific. In granularity-preserving shifts, the new vocabulary is constructible from the old one and vice versa. Granularity-preserving shifts are extremely hard to automate. An example from geometry is the shift from polar to cartesian coordinates.

In this paper, we study the class of granularity-coarsening shifts because (1) it is an important and useful class that includes abstractions and (2) we have systematic ways of defining the space of these shifts and domain-independent principles for traversing the space.

Definition 1 *Granularity-coarsening vocabulary shift: A granularity-coarsening vocabulary shift is a mapping $f : Vocab(T_2) \rightarrow Vocab(T_1)$ between vocabularies of definite Horn theories T_1 and T_2 , where $Vocab(T_2) = (c_2, r_2, l_2)$ and $Vocab(T_1) = (c_1, r_1, l_1)$. $f = (f_o, f_r)$ where*

1. *Object map: the function $f_o : c_2 \rightarrow Pwr(c_1)$ which maps object constants in T_2 to subsets of object constants in T_1 . $Pwr(x)$ denotes the power set of x .*
2. *Relation map: the function $f_r : r_2 \rightarrow L(Vocab(T_1))$ which maps relation symbols in T_2 to formulas in T_1 using definitions of the form $r(\vec{X}) \equiv p_1(Y_1), \dots, p_m(Y_m)$, where $r \in r_2$ is the relation symbol being defined with argument list $\vec{X}$, each p_i is a predicate symbol in r_1 (with possibly a leading negation symbol) with arity consistent*

with its argument list Y_i , and all variables in X occur among the variables in the Y_i 's. We require that relation symbols with standard interpretations (e.g., $=$) are not redefined by the relation map.

The mapping f_o on object constants can be represented by an indistinguishability relation $\approx$ on the constants in $Vocab(T_1)$. The object constants in $Vocab(T_2)$ are a subset of those in $Vocab(T_1)$ such that $f_o(a) = \{b \mid b \in Vocab(T_1), b \approx a\}$, where $a \in Vocab(T_1), Vocab(T_2)$.⁶ In essence, f_o allows us to carve up the set of objects assumed by T_1 into equivalence classes defined by the indistinguishability relation. In the shift from the pair (F_1, C_o) to the pair (F_2, C_o) , the map $f : Vocab(F_2) \rightarrow Vocab(F_1)$ consists of a 1-1 map from the object constants of F_2 to those in F_1 , and the relation map that defines max_ancestor as a formula in the vocabulary of F_1 ; viz., $f_r(\text{max_ancestor}(X, Y)) \equiv \text{ancestor}(X, Y) \wedge \neg \text{parent}(Z, X)$.

Definition 2 $Vocab(T_2)$ is coarser than or equal to ($\succeq$) $Vocab(T_1)$ if there exists a granularity-coarsening shift $f : Vocab(T_2) \rightarrow Vocab(T_1)$.

We will say that $Vocab(T_1)$ and $Vocab(T_2)$ are equivalent, denoted $Vocab(T_2) \equiv Vocab(T_1)$, if there exist granularity-coarsening shifts $f : Vocab(T_2) \rightarrow Vocab(T_1)$ and $g : Vocab(T_1) \rightarrow Vocab(T_2)$. $Vocab(T_2)$ is strictly coarser than ($\succ$) $Vocab(T_1)$ if and only if $Vocab(T_2) \succeq Vocab(T_1)$ and $Vocab(T_2) \not\equiv Vocab(T_1)$.

Proposition 1 $\succeq$ is a pre-order on a set S of vocabularies.

What prevents $\succeq$ from being a partial order is that it is not anti-symmetric. If relations in V_2 are constructible in terms of those in V_1 and vice-versa, then they are not necessarily identical relations. As a simple example, consider the fact that $\text{parent}(X, Y)$ and $\text{child}(Y, X)$ are interdefinable, yet are not identical.

Definition 3 *Vocabulary Space*: The space of vocabularies $VS(Vocab(T))$ generated by granularity-coarsening vocabulary shifts from a given theory T can be defined as

$$VS(Vocab(T)) = \{V \mid \exists f : V \rightarrow Vocab(T) \text{ satisfying Definition 1}\}$$

By Proposition 1, the set VS has the pre-order $\succeq$ defined on it. VS fails to be a lattice since $\succeq$ is not a partial order. However, like a lattice, VS possesses top and bottom elements. The top of VS is the vocabulary V_{top} such that $V_{top} \succeq V$, for all $V \in VS$. It has a single object constant and a unary relation constant whose extension is empty. V_{top} erases all distinctions between objects in the original theory T .

The set $VS/\equiv$ denotes the quotient of VS with respect to the $\equiv$ relation on vocabularies. This set is a lattice because equivalent vocabularies in VS are represented by a single canonical element. $VS/\equiv$ can be represented as a directed acyclic graph with a single root V_{top} and a single leaf $Vocab(T)$. The intermediate nodes in the graph are the interpolants V as in Definition 3 with equivalent nodes being represented canonically. Thus successors of any node V in the graph are strictly coarser vocabularies. Since the relations in the initial theory T are finite, and T has a finite number of objects, $VS/\equiv$

has finite depth. The branching factor is countably infinite because there are countably infinite f_r constructions in Definition 1. Our thesis is that minimization of distinctions, i.e., moving up toward the root in the graph $VS/ \equiv$ while preserving the goal relations yields theories with better computational properties.

Shift of vocabulary bias involves designing navigation policies in $VS/ \equiv$. To do this we need to be able to determine (1) whether a vocabulary shift preserves relations of interest (essential for evaluating coverage over a goal distribution), and (2) how a vocabulary shift is to be implemented. That is, given a theory T with vocabulary $Vocab(T)$ and a new vocabulary V created by the shift $f : V \rightarrow Vocab(T)$, how do we synthesize the theory T' that translates the formulas in T to the vocabulary V using the map f . This reconstruction step is essential for evaluating time and space costs.

Vocabularies are syntactic entities. In order to define what is preserved across a vocabulary shift, we introduce models of a function-free, definite Horn theory. The intended interpretation of a definite Horn theory is its least Herbrand model. For definite Horn theories which are consistent, there is a unique least Herbrand model (Lloyd, 1987). The Herbrand universe U of T is the set of all ground terms in T . The Herbrand universe U_2 of T_2 can be constructed from U_1 of T_1 via $f : Vocab(T_2) \rightarrow Vocab(T_1)$. Each term in U_2 stands for a collection of terms in U_1 .

$$U_2 = \{c \mid f(c) = \{c_1, \dots, c_m\}, c_1, \dots, c_m \in U_1\}$$

The Herbrand base B of T is the set of all ground positive literals that can be formed by using predicate symbols from T and ground terms from its Herbrand universe U . A Herbrand model of T is a subset of the Herbrand base. Given a formula ϕ with free variables $v_1, \dots, v_k$ and Herbrand model M_1 over a Herbrand universe U_1 , we say that $\models_{M_1} \phi[a_1, \dots, a_k]$ where v_i is assigned to $a_i \in U_1$ if $\phi[a_1, \dots, a_k]$ is true in M_1 . With this formula ϕ , we can associate the k -ary relation $\{ \langle a_1, \dots, a_k \rangle : \models_{M_1} \phi[a_1, \dots, a_k] \}$ which is the relation that ϕ defines in M_1 . We say that this relation is definable in M_1 since there is a formula ϕ that defines it. Given the least Herbrand model M_1 of T_1 and the relation map f_r , we can construct M_2 , the least Herbrand model of T_2 as follows

$$M_2 = \{p(b_1, \dots, b_k) \in B_2 \mid f(p) = \phi, \models_{M_1} \phi[a_1, \dots, a_k], a_i \in f(b_i)\}$$

Definition 4 *The least Herbrand model (LHM) of T_2 is constructed from the least Herbrand model of T_1 using $f : Vocab(T_2) \rightarrow Vocab(T_1)$.*

We can now show that there is no information gain in a granularity-coarsening shift. Intuitively, this is because every model of T_2 is constructible from a model of T_1 , but not necessarily vice versa.

Proposition 2 *Given a definite Horn theory T_1 and a granularity-coarsening shift $f : Vocab(T_2) \rightarrow Vocab(T_1)$, for every $\phi_2 \in L(Vocab(T_2))$ such that $T_2 \models \phi_2$, we have $T_1 \models_{f_L}(\phi_2)$, where f_L inductively extends the map f to formulas.*

What this proposition states is that for every formula entailed by T_2 , the translated formula in the vocabulary of T_1 holds in T_1 . So the theorems of T_2 are the translated

theorems of T_1 . No new theorems are added by the shift of vocabulary in the generation of the theory T_2 from T_1 .

We use Definition 4 to explicitly write the *reconstruction equation* that defines the least Herbrand model of the theory T_2 in terms of the least Herbrand model of T_1 and the vocabulary map f .

$$LHM(T_2) = Construct(LHM(T_1), f) \text{ using Definition 4.} \quad (1)$$

Generating solutions to the reconstruction equation above is called the implementation of a vocabulary bias shift.

3.3. Implementing vocabulary bias shifts

The implementation of a vocabulary bias shift yields a new theory. We need to calculate this theory because time and space costs can be evaluated only with respect to a specific theory and not a vocabulary. The abstract specification of the implementation operation is: Given T_1 and $f : Vocab(T_2) \rightarrow Vocab(T_1)$, solve the reconstruction equation 1 to find T_2 . The reconstruction equation does not identify T_2 uniquely. We take a simple propositional example to illustrate this point. Suppose the constructed model of T_2 is: $\{(f = true, g = false)\}$. A candidate encoding of T_2 is $\{f \wedge \neg g\}$. Another encoding is $\{f, f \Rightarrow \neg g\}$. We use the bias in the encoding of T_1 to restrict the space of possible reconstructions of T_2 .

Definition 5 A *granularity-coarsening shift* $f : Vocab(T_2) \rightarrow Vocab(T_1)$ from T_1 to T_2 is correct with respect to a goal distribution D if for every $\phi \in D$, $T_2 \models \phi \Rightarrow T_1 \models f_L(\phi)$, where f_L extends the map f to formulas.

The fact that f is correct does not guarantee full coverage over D , it only ensures that T_2 's answers are consistent with those of T_1 .

Definition 6 A *granularity-coarsening shift* $f_1 : Vocab(T_1) \rightarrow Vocab(T_o)$ from T_o to T_1 is better than the shift $f_2 : Vocab(T_2) \rightarrow Vocab(T_o)$ with respect to a goal distribution D and solver (T_o, C) , if there exist encodings T_1 and T_2 that satisfy the reconstruction equation such that $v(T_2) \geq_D v(T_1)$, where v is the value function on formulations.

Given this ordering between granularity-coarsening shifts, we can state the goal of a speedup learner precisely⁷: Find

$$\underset{f: V \rightarrow Vocab(T_o)}{\operatorname{argmax}}_{V \in VS(Vocab(T_o))} f$$

This is a specification of a global optimum in the space $VS(Vocab(T_o))$ of vocabularies. To identify local optima, we define four parametric operators that generate the vocabulary space starting with $Vocab(T_o)$. We sketch implementations of these operators below.

1. *Adding definable relations.* This transformation preserves information. A definable relation is constructed from the existing set of relations and objects. We denote the

operation $\text{add}(\tau, \phi)$, where ϕ is the defining formula for the relation symbol τ . If the rule defining τ is a Horn rule whose addition leaves the theory T definite (or safe and stratified for theories with negation), then we simply add the rule defining τ to T . Otherwise, we add τ as a primitive relation by precomputing it.

2. *Deleting relations.* This is an information-losing transformation. We denote the operation $\text{del}(\tau)$ where τ is a relation symbol. To delete the relation τ , we need to revise the theory so that no $\tau(\vec{x})$ can be derived in the theory. The smallest change (in the sense of set inclusion) to prevent the derivation of a $\tau(\vec{x})$ fact is implemented to minimize the effort expended for revision.

3. *Quotienting relations by defining equivalence classes on objects.* This is another information-losing transformation. Suppose we have relation $R : \mathcal{O} \times \mathcal{O}$. If we have an indistinguishability relation $\approx$ on $\mathcal{O}$, then the quotient $R/\approx$ is generally a smaller relation. We denote the operation $\text{quot}(\tau, \approx)$, where τ is the relation symbol denoting R .

If τ is primitive, the quotient set τ_{approx} is extensionally calculated. To quotient a defined or computed relation τ , we explicitly calculate canonical elements from the $\approx$ relation (a forward chaining computation) and store it in the binary canonical relation. The quotient relation is then defined as $\tau_{\text{approx}}(X) : \neg\tau(Y), \text{canonical}(X, Y)$.

4. *Converting computed relations to primitive relations and vice versa.* This is an information-preserving transformation. In problem solvers where the cost of solving primitive and computed relations is sufficiently different, this distinction is important. We denote this operation $\text{convert}(\tau, l)$ where l is the new label for relation symbol τ .

To convert a computed relation τ to a primitive one, we forward chain on the theory, generating all derivable τ facts, and store them explicitly. This technique is used for materializing finite relations only. To convert a primitive relation to a computed one, we find a defining formula whose extension is the primitive relation.

Proposition 3 *Every granularity-coarsening shift $f : \text{Vocab}(T_2) \rightarrow \text{Vocab}(T_1)$ can be constructed from the four operators: $\text{add}(\tau, \phi)$, $\text{del}(\tau)$, $\text{quot}(\tau, \approx)$ and $\text{convert}(\tau, l)$.*

Each operator application from $\text{Vocab}(T_o)$ generates an element in the vocabulary space $VS(\text{Vocab}(T_o))$. The specific moves needed to generate $\text{Vocab}(F_2)$ and $\text{Vocab}(F_3)$ from $\text{Vocab}(F_1)$ are shown in Figure 2.

Definition 7 *The vocabulary $\text{Vocab}(T) \in VS(\text{Vocab}(T_o))$ (resp. the granularity-coarsening shift $f : \text{Vocab}(T) \rightarrow \text{Vocab}(T_o)$) is a local optimum for the goal distribution D and the value function v if there is no $\text{Vocab}(T')$ constructible from $\text{Vocab}(T)$ by the four parametric operators such that $v(T') \geq_D v(T)$.*

We can now define the hill-climbing procedure in Table 1 for finding local optima in the vocabulary bias space. What the procedure leaves open is the instantiation of

Figure 2. A portion of the space of vocabulary shifts searched by our speedup learner. Paths that generate solvers (F_2, C_o) and (F_3, C_o) are shown. Relation names in bold are primitive relations, while relation names in italics are computed. **p** stands for the parent relation, *sf*, *anc*, **ma** denote the samefamily, ancestor and maximal ancestor relations respectively. The space has a countably infinite branching factor, and a large, though finite, depth.

procedure find-local-optimum $((T_o, C_o), v, D)$

1. $(\text{Top}, \text{BestVal}) = ((T_o, C_o), v((T_o, C_o)))$
2. Successors = $\{[(T, C), v((T, C))]\mid (T, C) \text{ such that } \text{Vocab}(T) \text{ generated from } \text{Vocab}(\text{Top}) \text{ by one of the four parametric operators}\}$.
3. **if** $\text{BestVal} \geq_D x$ for every $[S, x] \in \text{Successors}$
then stop and output Top.
else $(\text{Top}, \text{BestVal}) = [S, x]$ such that $x \succeq_D y$ for every $[S, y] \in \text{Successors}$.
Go to 2.

Table 1. Hill-climbing search for a local optimum in the vocabulary bias space. v is the evaluation function for solvers and D the goal distribution.

the parametric vocabulary shift operators. The next subsection details a meta-theoretic approach to the selection of instantiated operators in the vocabulary bias space. While the procedure is committed to taking single steps in the vocabulary space to generate new formulations, in practice the expensive evaluation process necessitates taking directed macro-steps in the space. This issue is discussed in greater detail in Section 4.3.

3.4. Justifying vocabulary bias shifts

Our approach is to provide a unifying framework and a set of concepts that allow *declarative specification* of the knowledge that is required for automatic shift of vocabulary bias. This makes it possible to evaluate a solver (T, C) along the three dimensions of coverage, space and time costs and to determine how to modify $Vocab(T)$ to improve the value function v .

We write *irrelevance* explanations for the role of each vocabulary item in the computation of the goal distribution.⁸ An irrelevance explanation proves that certain distinctions in a theory are not logically necessary for the solution of a given class of goals. We say that an irrelevance claim holds in the meta-theory if the distinction specified by the claim is irrelevant.

In the kinship problem, observe that the `parent` relation in the `samefamily` computation in F_1 plays two roles. Each `parent(X, Y)` fact (1) establishes first level ancestry `ancestor(X, Y)` through rule 1, and (2) establishes the set $\{\text{ancestor}(X, Z) \mid \text{ancestor}(Y, Z)\}$ derivable in (F_1, C_o) through rule 2. We can represent this fact as follows:

$$\begin{aligned} derivable_from(\text{parent}(X, Y), F_1) = & \{ \text{ancestor}(A, B) \mid \\ & (X = A, Y = B, \text{parent}(X, Y) \in F_1) \\ & \vee (X = A, F_1 \vdash_{C_o} \text{ancestor}(Y, B)) \} \end{aligned}$$

where $T \vdash_C f$ denotes that f is derivable from a Horn theory T with control strategy C . We can eliminate an element of the `parent` relation, if we have derived from it all that is *essential* to the `samefamily` computation. $derivable_from(P, (T, C))$ consists of the set of facts (other than P itself) in T that directly require P for their derivation. This set is not deductively closed.

$$\begin{aligned} derivable_from(P, (T, C)) = & \{ Q\theta \mid (Q : \neg C_1, \dots, C_m) \in T, \\ & \theta = mgu(P, C_i) \text{ for some } i, T \vdash_C Q\theta \} \end{aligned}$$

We now define the ternary relation $irrelevant(f, g, (T, C))$ on facts f with respect to goals g in (T, C) . By defining irrelevance at the level of facts (elements of a relation) instead of on entire relations, we can prove parts of a relation to be irrelevant.

Definition 8 A fact f is irrelevant to a goal g with respect to the pair (T, C) of Horn sentences T interpreted by C , written $irrelevant(f, g, (T, C))$, if there exists $T' \subseteq T$ such that $T' \vdash_C g$ and $T' \not\vdash_C f$. We call T' the reduction of T by the claim $irrelevant(f, g, (T, C))$ and denote it as $reduce(T, irrelevant(f, g, (T, C)))$.

The irrelevance claim is a statement about the set of proofs of g in T ; in particular, it states that it is possible to construct proofs of g in T without establishing f . An irrelevance claim is an *intensional* description of this property in the space of proofs. The claim $irrelevant(f, g, (T, C))$ sanctions the simplification of the theory T to T' as described in Definition 8.

It is useful to compare our notion of irrelevance with the Anderson and Belnap notion of relevance. The two notions of relevance and irrelevance are complementary; one

expresses a dependence between two facts and the other captures a one-sided lack of dependence (g being not dependent on f). However, the normal use of an irrelevance statement is to modify T while preserving g , the normal use of a relevance statement is to infer restrictions on g given f : thus the inferences they sanction are not duals.

In the kinship example, we can state the irrelevance of the entire `parent` relation as follows.

$$\begin{aligned} \mathbf{IC1} : & \forall X, Y, M, N \text{ derivable_from}(\text{parent}(X, Y), F_1) \subseteq F_1 \\ & \Rightarrow \text{irrelevant}(\text{parent}(X, Y), \text{samefamily}(M, N), (F_1, C)) \end{aligned}$$

This *conditional* irrelevance claim states that if the condition on the left hand side is met, then it is possible to derive all `samefamily` goals in a new theory without using any `parent` facts. To check this irrelevance claim using Definition 8, we augment F_1 minimally (in the sense of subset inclusion) to make the condition on the left hand side of the claim true. For example, if the condition is of the form $x \in T$ or $x \subset T$, we add x to T . Then the subset construction is applied to generate the reduced T' .

We now relate computational properties of $T = \text{reduce}(F_1, \mathbf{IC1})$ to that of F_1 . Intuitively, the reduction of F_1 by $\mathbf{IC1}$ results in the removal of the `parent` relation and the addition of the entire transitive closure of the `parent` relation to F_1 . This step increases the space cost. Its impact on time cost is a function of the distribution of `samefamily` goals. On average, we expect time cost on successful goals to reduce, since they are solved by looking up rather than by inferring `ancestor` facts.

We can improve the space cost of the reduced theory generated above by recognizing that the identity of the common ancestor in the last rule of `samefamily` is irrelevant. The corresponding irrelevance claim is:

$$\begin{aligned} \mathbf{IC2} : & \forall X, Y, Z, M, N \text{ ancestor}(X, Y) \in T \wedge \text{ancestor}(Y, Z) \in T \wedge \text{ancestor}(X, Z) \in T \\ & \Rightarrow \text{irrelevant}(\text{ancestor}(Y, Z), \text{samefamily}(M, N), (T, C)) \end{aligned}$$

Reduction of T by $\mathbf{IC2}$ yields the desired restriction on the `ancestor` relation which serves as the definition of the `max_ancestor` relation.

$$\begin{aligned} \text{ancestor}(Y, Z) \in \text{reduce}(T, \mathbf{IC2}) & \equiv \\ \text{ancestor}(Y, Z) \in T \wedge \neg \exists X (\text{ancestor}(X, Y) \in T \wedge \text{ancestor}(X, Z) \in T) & \end{aligned}$$

The above expression states that if an `ancestor(Y, Z)` fact survives the reduction process, then the LHS of $\mathbf{IC2}$ must not hold. Therefore, there cannot be an x such that `ancestor(X, Y) ∧ ancestor(X, Z)`. Using transitivity of `ancestor`, we can eliminate the last conjunct (`ancestor(X, Z)`) to yield the following intensional description of the remaining `ancestor` facts.

$$\begin{aligned} \text{ancestor}(Y, Z) \in \text{reduce}(T, \mathbf{IC2}) & \equiv \\ \text{ancestor}(Y, Z) \in T \wedge \neg \exists X \text{ ancestor}(X, Y) \in T & \end{aligned}$$

The structure of the irrelevance explanations yields candidates for the `del`, `add` and `convert` operators directly. If the distinction between objects is irrelevant to a computation, the irrelevance claim that describes it will define the appropriate $\approx$ relation for

the `quot` operator. To automatically instantiate the parametric operators in the space of vocabulary bias shifts we adopt the following scheme.

1. Generate those irrelevance claims for the goal class that improve computational properties of the formulation. Each irrelevance claim is an explanation for a specific simplification or tradeoff between coverage and computational efficiency. The set of irrelevance claims specifies a set of trajectories through the vocabulary bias space.
2. Reduce the theory by the generated irrelevance claims to produce simplifications that satisfy the reconstruction equation (equation 1).

In the next section, we describe algorithms for these two steps and present experimental results from the domain of reachability computations.

4. Algorithms for automatic shift of vocabulary bias

We present two generation schemes – one instantiates irrelevance schema that capture standard patterns of irrelevance in definite Horn theories, the other examines the computation graph of the goal scheme and identifies instances of irrelevant computation. We show that no complete method for generation of irrelevance claims can exist. We present two reduction schemes for Horn theories – one reduces a theory with respect to unconditional irrelevance claims; the other is an incremental procedure for reducing conditional irrelevance claims. We finally assemble all components together to create a system that automatically shifts vocabulary bias. We present experimental results obtained by running the system on reachability computations in Gridworld (Tambe & Rosenbloom, 1989), (Cohen, 1990)).

4.1. Generating irrelevance claims

4.1.1. Theoretical methods: irrelevance schemas

We use Definition 8 to identify standard patterns of irrelevance in definite Horn theories. For instance, if we derive every fact that directly requires x using the *derivable_from* relation introduced in the previous section, then we can throw x away. This scheme can be stated as follows.

$$\mathbf{S1} : \text{derivable_from}(P, G, (T, C)) \subseteq T \wedge P \not\subseteq G \Rightarrow \text{irrelevant}(P, G, (T, C))$$

Proposition 4 *S1 satisfies Definition 8.*

As an instance of the use of the schema, consider the derivation of **IC1** from F_1 by instantiating P with `parent(X, Y)`, T with F_1 , and G with `samefamily(M, N)`.⁹ The set *derivable_from*(`parent(X, Y)`, (F_1, C_o)) can be calculated from the general definition of *derivable_from*. The irrelevance claim **IC2** can also be derived as

an instance of **S1** using rule 2 of F_1 . This derivation requires the calculation of $\text{derivable_from}(\text{ancestor}(Y, Z), (F_1, C_o))$ and yields the LHS of IC2 .

If we can generate a complete set of schemas like **S1** to identify all irrelevancies in a formulation, we can theoretically derive all irrelevance claims for a (T, C) pair and a goal class g . This amounts to detecting all fact classes f such that T' (which is the revision of T to eliminate f) and T are g -equivalent. (Shmueli, 1987) showed that g -equivalence is an undecidable property implying that we can never write down a complete set of irrelevance schemas. Therefore we turn to heuristic methods for generating irrelevance claims.

4.1.2. Empirical methods: transforming computation graphs

We can extract several classes of irrelevance facts directly from a C-ordered search graph S of a computation. The computational effort (time cost) of solving g in (T, C) is proportional to the size of S . The general idea is to transform S by operations that preserve answers to the goal scheme while reducing the size of S .

The C-ordered search graph S of a goal scheme G in the theory T , is a representation of the search space of proofs of g in (T, C) . Each node in the graph is a set of subgoals ordered by C . The variables in subgoals are annotated to indicate whether they are bound or free. This annotation is done automatically by an algorithm in Appendix II that generates the graph from (T, C) and g . Each single arc denotes a one step inference using a rule in T . A proof of G in (T, C) is a path in the C-ordered search graph that starts at the root and terminates in a leaf node with the empty clause. Note that in Figure 3, there are several paths from the root that end in the empty clause. Since the underlying theory is recursive, there are doubly-lined back edges in the graph linking sets of subgoals that are identical up to variable renaming. Note that in the construction of the C-ordered search graph S , we do not expand literals corresponding to primitive relations. Cycles (doubly-lined back edges in S) are traversed exactly once.

There are two generic size-reducing operations on a C-ordered search graph S : reduction of depth, and reduction of effective branching factor. To reduce depth, we eliminate intermediate computation along branches in S . This is the operation underlying the formation of macrops (Korf, 1987), macro rules (Keller et al., 1986) and chunks (Laird et al, 1987). To reduce the effective branching factor we prune unsuccessful branches in S .

Definition 9 A computational step in a C-ordered search graph S (edge in S from a node n_s to node n_e) is irrelevant with respect to a goal g (root node of S) if one of these conditions holds. l is the leftmost literal in n_s .

1. its result is unneeded. Substitutions generated by the solution of l do not contribute to a solution of the root node.
2. its result is already known. Substitutions for l generated at node n_s are contained in those generated in n_f that occurs earlier in a depth-first traversal of S .

Figure 3. The C-ordered search-graph generated by a depth-first backward chainer with the standard control strategy C_o [rules from top to bottom and antecedents from left to right] for the $\text{samefamily}(X^b, Y^b)$ goal in the formulation F_1 . anc and par are abbreviations for ancestor and parent respectively. The double lines link subgoals that are identical up to variable renaming.

Figure 4. Transforming computation graphs by thread variable elimination. s is a computed relation, p, q, r are primitive relations. The C-ordered SLD graph S of the original theory is shown on the left. In the middle is a transformation that shortcircuits S by eliminating the thread variable Y with the introduction of the new primitive relation $t(X, Z)$ defined to be $p(X, Y) \wedge q(Y, Z)$. The irrelevance claims that implement the transformation are shown below it. The graph on the right is another transformation to S obtained by eliminating the thread variable Z .

For forward-chaining problem solvers, deriving unneeded facts (condition 1 in definition 9) is a significant source of irrelevant work. For backward-chaining solvers, inferring the same facts more than once (condition 2 in definition 9) results in irrelevant computation.

Detecting these conditions requires considerable analysis (both static and run-time) of the computation graph. In the worst case, it requires finding all the solutions to every subgoal that occurs in S . We implement a very special case of this definition, called *thread variable elimination*, that is syntactically cheap to detect. Figure 4 presents a simple example of thread variable elimination in a non-recursive, definite Horn theory. Thread variable elimination constructs shortcuts in the computation graph by eliminating intermediate nodes with large branching factors.

Definition 10 *A variable in the antecedent of a rule r is a thread variable if it is a shared variable that does not occur in the consequent of r .*

Thread variables are syntactic signals for a special case of irrelevant computation. The algorithm in Table 2 generates irrelevance claims using thread variable elimination as well as specialized pruning of unsuccessful branches in the C-ordered search graph. It aggregates a conjunction with shared variables into a single pre-computed relation. It deletes the relation corresponding to a component conjunct as long as that conjunct does not occur separately along another successful path in the tree. The pruning of unsuccessful branches generates a claim that is a conjunction of all literals reduced along that path. The thread variable elimination technique allows for the exploration of the tradeoff between space and time costs in a theory by precomputing various definable relations in the theory. The technique can be recursively applied to any state-space

procedure *generate*($g, (T, C)$)

1. S = the C-ordered search graph for g using (T, C) ;
2. Non-deterministically choose:
 - a. Shortcutting:
for every path p_s from the root of S to an empty leaf do;
 Identify a node n_s with thread variables V
 with at least one $v \in V$ occurring in leftmost literal l_0 in n_s .
 No $v \in V$ should occur in the root of S .
 $C(s)$ are the constraints on V gathered from the root to node n_s .
 If l_0 shares the thread variables in V with literals $l_1, \dots, l_m$ in n_s
 generate the new relation $t(\vec{X}) = l_0 \wedge \dots \wedge l_m$
 where $\vec{X} = [\cup_{i=0}^m \text{vars}(l_i)] - \{V\}$.
 The irrelevance claims are $t(\vec{X}) \in T \wedge C(s) \Rightarrow \text{irrelevant}(l_i, g, (T, C))$
 for l_i that do not unify with a literal on another successful path in S .
 - b. Pruning:
for every path p_f from the root of S to a non-empty leaf node do;
 construct the irrelevance claim $C(p_f) \Rightarrow \text{irrelevant}(\bigwedge_{l \in n} l, g, (T, C))$,
 for every node n in p_f , where $C(p_f)$ consists of constraints on all
 the variables occurring in p_f .

Table 2. The empirical irrelevance claim generator. It examines the C-ordered search graph S of g in (T, C) and transforms it to S' by (a) elimination of intermediate steps along a successful branch, and (b) pruning unsuccessful branches. Each transformation of the C-ordered search graph generates a set of irrelevance claims. The reduction of T by the claims produces a new (T', C) that has S' as its C-ordered search graph for g .

search problem formulated as a reachability computation.¹⁰ The vocabulary shift in the kinship example is an instance of thread variable elimination in a recursive theory.

Proposition 5 *Algorithm generate($g, (T, C)$) is sound but not complete.*

It can easily be shown that the irrelevance claims generated by the algorithm are correct using Definition 8 and that irrelevant computation is eliminated as per Definition 9 by these claims. Unfortunately, we cannot detect all cases of irrelevant computation in recursive theories (since it is an undecidable property), so we cannot ever produce a complete method. Our experiments, detailed in Section 4.3, show that the algorithm generates all the useful irrelevance claims for transitivity-style recursions.

4.2. Reducing theories by irrelevance claims

Once we have an irrelevance claim of the form $\text{irrelevant}(f, g, T)$, we can reduce a theory T by that claim to generate a new theory T' which satisfies Definition 8. The theory T' has a new relational basis since f is either completely or partly eliminated. We present an algorithm for reduction of Horn theories and prove that it produces correct solutions to the reconstruction equation 1.

Consider a theory

$$T = \{g(X):-a(X). \quad g(X):-b(X). \quad b(1).\}$$

with two primitive relations a and b and a goal relation $g(Y^f)$. We can easily show that $irrelevant(a(X), g(Y^f), (T, C))$ since any single solution to g suffices.. There are two possible reduction procedures for T with this respect to this irrelevance claim: (1) remove every clause in T in which a literals occur positively; thus all ground facts about a are eliminated from T ; or (2) remove every clause in T in which a literals occur negatively; this process discards the rule $g(X):-a(X)$. Both options are “minimal” in the sense of set inclusion. Both prevent derivation of g via a facts – one by eliminating all ground a facts to render the rule $g(X):-a(X)$ useless, and the other by eliminating the rule itself while keeping all the ground a facts. Thus the reduction of T by this irrelevance claim is not unique. In our implementation, we generate both and pick the reduction that is better under the value function v .

We now illustrate issues in the elimination of computed predicates, via an example. Consider the non-recursive definite Horn theory

$$T = \{f(X):-p(X), q(X). \quad f(X):-r(X). \quad g(X):-f(X). \quad g(X):-h(X). \quad h(a).\}$$

where p, q and r are primitive relations, f is the computed relation to be eliminated and $g(Y^f)$ is the goal relation. We can see that $irrelevant(f(X), g(Y^f), (T, C))$, since we are interested in any one solution to the predicate g and there is a derivation for it via h . To reduce the theory T to eliminate f , we need to cut every successful path in the computation graph of g that contains a literal that unifies with f . However, each path can be cut at one or more points, yielding a combinatorial number of revision possibilities. One option is to eliminate every rule in which f occurs positively. This breaks the derivation of f at the very last step, since all the antecedents to establish f remain derivable. A second option is to recursively eliminate some antecedents of every rule in which f occurs as a consequent. In this example, we can eliminate r and p , or r and q to obtain the largest subset of T that does not entail f . If p (or q or r) had been a computed relation, we can recursively trace its primitive support set. We check (using the procedure verify in Table 4) that the recursive removal does not eliminate literals that are essential for solving the goal.

Proposition 6 *The reduction procedure in Table 3 satisfies its specifications. That is, upon termination it produces $T' = reduce(T, irrelevant(f, g, (T, C)))$ such that $T' \not\models_C f$ and $T' \vdash_C g$.*

We will say that the revision T_1 is *better* than the revision T_2 of T with respect to an irrelevance claim if $v(T_1) >_D v(T_2)$. To ensure that the best revision is generated, we need to examine all revisions. Since the number of revisions is exponential in the size of T , we can expect the problem of generating the best revision to be computationally intractable.

Proposition 7 *The problem of obtaining the least-cost (with respect to a value function v) revision of T with an irrelevance claim is NP-hard.*

To implement the shift suggested by a conditional irrelevance claim, we first augment T by the propositions specified in the condition, provided they are members of the deductive

procedure *reduce*($T, \text{irrelevant}(f, g, (T, C))$)

Input: a Horn theory T and a correct irrelevance claim with respect to g

Output: a Horn theory T' such that f is not derivable in T' and g is.

Case 1: f is atomic:

Non-deterministically generate one of:

- a. $T' = T - \{c \mid c \in T, f \text{ occurs positively in } c\}$.
- b. if f is primitive, then $T' = T - \{c \mid c \in T, f \text{ occurs negatively in } c\}$.
- c. if f is derived, then build a list L of rules with heads that unify with f .

Initialize T' to be T .

for each clause $c = a_1, \dots, a_m \Rightarrow f$ in L ,

 If *verify*(*irrelevant*($a_1 \wedge \dots \wedge a_m, (T', C)$)) then

 set T' to be *reduce*($T', \text{irrelevant}(a_1 \wedge \dots \wedge a_m, g, (T', C))$)

Case 2: f is a conjunction $f_1 \wedge f_2$:

Non-deterministically generate one of:

- a. if *irrelevant*($f_1, g, (T, C)$) then $T' = \text{reduce}(T, \text{irrelevant}(f_1, g, (T, C)))$
- b. if *irrelevant*($f_2, g, (T, C)$) then $T' = \text{reduce}(T, \text{irrelevant}(f_2, g, (T, C)))$

Table 3. The reduction procedure *reduce*($T, \text{irrelevant}(f, g, (T, C))$).

procedure *verify*($f, g, (T, C)$)

f and g are positive literals. All variables in f are annotated as free.

1. S = the C-ordered search graph for g using (T, C) ;
2. $P = \{p \mid p \text{ is a path in } S \text{ from the root to an empty leaf with cycles traversed exactly once}\}$;
Associate with every $p \in P$ a set $C(p)$ – which is a list of all left-most literals in a node in p with an out edge labeled a (these are literals solved by lookup, unification or arithmetic).
3. Case 1: All variables in g are annotated as free:
 if there exists a path $p \in P$ in which a literal with predicate f does not occur
 then return $C(p)$
 else return false
4. Case 2: Some or all of the variables in g (the list $\vec{X}$) are annotated as bound:
 Compute s_f , the set of all solutions to g via the paths $p \in P$ on which f lies.
 Compute s_o , the set of all solutions to g via other paths in P .
 if $s_f \subseteq s_o$ then return true else return $\vec{X} \neq s_o - s_f$

Table 4. Algorithm *verify*($f, g, (T, C)$) implements the predicate *irrelevant*($f, g, (T, C)$) for Horn theories T . It works for atomic f 's and g 's only. It returns unification and arithmetic constraints under which f is irrelevant to g . For a more detailed explanation of this procedure as well as a scheme for handling non-atomic f 's, see Appendix III.

closure of T . As an example, consider reduction by **IC2** of the formulation F_1 , after reduction by **IC1**. When we find $\text{ancestor}(X, Y)$ and $\text{ancestor}(Y, Z)$ in the database, we add $\text{ancestor}(X, Z)$ and delete $\text{ancestor}(Y, Z)$, for each instantiation for x, y and z in the database. The incremental rewiring of the ancestor relation terminates when all ancestry links are one step long. This process creates the max_ancestor relation extensionally. The conditional irrelevance reduction procedure has two important properties.

1. *Order of reduction:* When there is more than one irrelevance claim, there is a choice to be made regarding the order of reduction. When the claims are independent, then the order is irrelevant. In general, the reduction procedure does not satisfy the Church-Rosser property. That is, the final formulation is dependent on the order in which the reductions are applied.
2. *Termination of reduction:* The fact that checking termination reduces to the halting problem imposes fundamental limits on inferring it in the general case. However, in most instances, we can determine termination by noting that relations are finite and acyclic. The kinship example is one such case.

4.3. Experimental results

The vocabulary shifter that we implemented has three major components: (1) Irrelevance claim generator with schema-based generation and thread variable elimination, (2) Formulation reducer for both unconditional and conditional irrelevance claims, (3) Formulation evaluator that samples the goal distribution and estimates coverage and average time cost (real number) to within user-specified error (ϵ) and confidence (δ) parameters.¹¹ The formulation evaluation function v used is a linear combination of coverage, time and space cost. We experimented with two different policies for sequencing the reduction and evaluation stages. A formulation can be evaluated after reduction by one or all of the generated irrelevance claims. The latter policy corresponds to taking directed macro-steps in $VS(\text{Vocab}(T_o))$. In the kinship example explained in this paper, we illustrated this scheme. In the Gridworld domain considered next, the incremental policy illustrated in Table 1, was employed successfully.

We performed scaleup studies of our shifter by taking a parametric domain in which we can easily vary the complexity of problems. The domain considered is Gridworld introduced in (Tambe & Rosenbloom, 1989) and studied in the context of learning control rules for speedup in (Cohen, 1990). The Gridworld is a square n by n grid of points $p_i, 1 \leq i \leq n^2$, represented by the $\text{conn}(X, Y)$ relation, which records 1-step connections in the grid. conn is a symmetric predicate. The $\text{reachable}(X, Y, D)$ predicate is recursively computed from conn using the definite Horn theory shown below, and represents the fact that there is a path of length D that connects x and y in the grid. The predicate iterate_reachable implements an iterative-deepening strategy to compute a path between two points by attempting to find paths of successively increasing lengths from 1 to a specified maximum. The goal class is $\text{goal}(X^b, Y^b, \text{Max}^b)$ where x and y are grid points and Max is the maximum depth to which we wish to look for a path between these

grid points. The complexity of this goal class varies exponentially with the grid size n ; a fact reflected in Figure 5.

```

reachable1(A,B) :- conn(A,B).
reachable1(B,A) :- conn(A,B).
reachable(A,B,1) :- reachable1(A,B).
reachable(A,B,N) :- N > 1, N1 is N-1, reachable1(A,C),
                      reachable(C,B,N1).

iterate_reachable(A,B,Start,End,_) :- Start > End, !.
iterate_reachable(A,B,Start,End,Start) :-
    reachable(A,B,Start).
iterate_reachable(A,B,Start,End,D) :-
    Start1 is Start+1, iterate_reachable(A,B,Start1,End,D).
goal(N,M,End) :- iterate_reachable(N,M,1,End,D).
```

We used the technique of thread variable elimination to generate the one irrelevance claim for this domain. If $\text{conn}(X,Y)$ and $\text{conn}(Y,Z)$ are present, we eliminate $\text{conn}(X,Y)$ by adding all $\text{conn}(X,Z)$ facts such y is one-step connected to z . This irrelevance claim applies to all grid sizes. We generated the claim using both the empirical and the schema-based irrelevance claim generation method. In all cases, the reducer rewired the conn relation so that all points in the grid are 1-step connected to a common point. The identity of this common point varied from run to run, but we consistently obtained this reduction under several sampling strategies for goals in the distribution, as well as value functions for evaluating formulations. It is important to note that no change in the control strategy alone can simulate this shift of representation of the underlying grid. We now report the results of several experiments we conducted.

1. *Performance of the local optimum generated by reduction, under varying grid sizes:* We generated reductions of the given formulation by the irrelevance claim described above, under grid sizes $2 \leq n \leq 7$. We computed reachability over 1000 point pairs (x,y) drawn from a uniform distribution on an n by n grid, using the original as well as the reduced theory. All problems were allowed to run till completion. The timing measurement for each goal in the problem set was averaged over 10 trials and the Wilcoxon signed rank test was used to judge the significance of the speedup. The reduced theory outperformed the original theory at a significance level of 0.1%. The actual differences in times between the original and the learned theory are shown in Figure 5.
2. *Performance of the local optimum generated by reduction, under different goal distributions:* We checked if the local optimum produced was robust under the choice of different goal distributions. We generated 1000 (x,y) point pairs from an n by n grid where the x 's were restricted to be chosen randomly from between 1 to n points, simulating single-source problem distributions to the random distribution of the previous set of experiments. In all cases ($2 \leq n \leq 7$), the formulation converged to a set of facts where each grid point was 1-step connected to a common grid point.

Figure 5. Performance of the original domain theory and the new theory generated by irrelevance minimization as the complexity of Gridworld is scaled up.

3. *Performance of the extensional reducer:* An important issue is the time taken for the reduction process. Each iteration of the reduction process involves checking whether implementing that reduction yields a speedup. The formulation evaluator efficiently samples the goal distribution to make this test with high confidence with relatively few samples (several 10s of problems). In Figure 6 we show the number of iterations taken for convergence to a local optimum, and the decrement in average time over the goal distribution with each iteration. Surprisingly, as the grid size increases, the number of local minima (where time decrement is 0) that the reducer finds decreases substantially. Our reducer uses the policy of implementing the “best” reduction, until there are no more reductions to process. This policy allows it to continue even if it hits a local minimum in the formulation space until all irrelevancies specified by the claims are eliminated.

4.4. Discussion

The kinship example in this paper is a reformulation of an equivalence relation as a partition; this is a standard transformation taught to every computer scientist. The method of irrelevance minimization explains how this transformation can be derived from more general considerations. The Gridworld transformation with a different formal structure (symmetric base relation) also employs an irrelevance claim very similar to the one used in the kinship example. The reduction generated is novel compared to the optimizations produced by other speedup learning methods. We have only applied our techniques to reachability computations in graphs. Our generation and reduction techniques are sufficient to learn interesting relations and aggregations for such problems. While our

Figure 6. Performance of the gradient descent bias shift algorithm. All times are scaled by 10 for $N=3$ so that it can be displayed alongside the data for $N=4$.

theoretical framework applies to more general problems, we do not have computationally tractable algorithms for restructuring other recursive computations. The algorithms described here work for only computations specified as definite (as well as stratified) Horn theories. However, Webber (Webber, 1993) has extended the theoretical framework described in this paper to the computational paradigm of functional programming. Starting from a basic definition of an irrelevant computational step (Definition 9), he has derived all known compiler optimizations for functional languages.

The specific algorithms presented here generate subsets of the deductive closure of the original theory T_o .¹² Since deletion of parts of relations is allowed, we also generate specializations. Our evaluation functions only take the quality of the final product into account and not the cost of the procedure for obtaining them. So far, the cost has been fairly small in comparison with the performance advantage gained with the use of the end products of the reduction. The next step in our work is to fold in the cost of navigating in $VS(Vocab(T_o))$ by posing a constrained optimization problem of detecting local optima within specified resources allocated to the optimization procedure.

5. Related work and conclusions

5.1. Related work

The difficulties of automating vocabulary shifts has deterred progress in this area for a long while. In 1980, Korf (Korf, 1980) characterized the nature of the information transformations that occur during shifts of vocabulary. He classified transformations as information-preserving, information-losing and information-gaining. This classification

is similar to the types of granularity shifts described here. However, our classification, which is more abstract, is in the space of ontologies of theories, and not on the transformations themselves. Korf specified a collection of rewrite rules on theories to define a space of possible shifts. His theory explained vocabulary shifts in the Tower of Hanoi puzzle as well as some examples from floor planning. The rewrite rule approach was also followed by Jack Mostow who designed a program in 1981 that reformulated advice in the game of Hearts. He devised 400 domain-specific as well as general transformation rules to define the space of reformulations, which included shifts of vocabulary. The main disadvantage of this approach was that there were many rewrite rules and the problem of deciding the order in which to apply them was formidable. Another disadvantage was that there was no systematic way of generating the rewrite rules from more global considerations. Our meta-theoretic approach, on the other hand, reasons from first principles about wasted computation, and derives the appropriate syntactic transformations to make a theory more efficient.

In 1983, Utgoff (Utgoff, 1986) attacked automating shifts of vocabulary bias in the inductive context with the Lex (Mitchell et al., 1982) problem solver. He proposed a method called *back-propagation* to refine the concept of *integer* to *odd-integer* to make an integration heuristic expressible. The chief insight in this work was the explicit use of problem-solving goals to guide the addition of new vocabulary. Our approach builds on this insight by predicating irrelevance claims with respect to goal classes. The addition of new vocabulary to an inductive system to make prediction efficient was studied by Fu & Buchanan (Fu & Buchanan, 1985). This work proposed two general heuristics and was tested empirically in the domain of medical diagnosis. It is an interesting open problem to derive the general principle from which these heuristics can be generated. In 1986, Richard Keller (Keller, 1987) introduced a scheme for the addition of new vocabulary items into the Lex solver to make the solution of special query classes efficient. His contribution was the use of empirical methods for testing the efficacy of a given representation for a class of goals. Our scheme for evaluating new formulations builds on this work and augments it with the use of sampling strategies with statistical guarantees.

More recent work in this area focus on very special classes of deductive and inductive abstractions. Riddle (Riddle, 1988), building on work done by Saul Amarel in the 1960s (Amarel, 1968), partially automated the formation of deductive abstractions of the missionaries and cannibals problem. Some of Riddle's transformations (geared for solvers based on problem-reduction) can be derived by reasoning about irrelevance of distinctions in a formulation. Lowry (Lowry, 1988) developed methods based on the theory of abstract data types to synthesize special-purpose algorithms from specifications. Muggleton (Muggleton, 1988) devised a scheme that introduced new predicates to make compact generalizations of instances. Flann (Flann, 1988) reformulated theories expressed in structural terms into functional terms to make a recognition task more efficient. Schlimmer (Schlimmer & Proca, 1988) generated functionally useful classes by induction from problem solving traces. Knoblock's method (Knoblock et al, 1991) automatically generates abstraction of operators in Strips planning by ordering operator preconditions. All these methods use special knowledge about the problem domain, and the solver to generate theories based on new abstract vocabularies. They can be viewed

as compilations of general principles of vocabulary shifts, including the irrelevance principle described here. An important open question is to derive the specific vocabulary shifts by reasoning about transformations to the computation of the goal class guided by general principles of shifts of vocabulary bias.

5.2. *Conclusions*

Research in automating shifts of vocabulary bias in speedup learning is still exploratory in nature. The main contribution of this paper is a conceptual framework in which relevant questions about shifts of vocabulary bias for computational efficiency can be phrased and answered.

We introduced the need for meta-theoretic explanations of the role of a vocabulary item for justified bias shifts. We discussed a special class of such explanations, based on irrelevance. Irrelevance explanations are unique because they prescribe changes to a theory's vocabulary that impact its computational properties. The crux of our paper is the use of the irrelevance principle to transform these irrelevance explanations into prescriptions for eliminating certain relations and adding new definable ones.

The chief algorithmic result is the development of (i) a hill climbing search of a well formulated bias space aided by irrelevance claim generators and formulation reducers, (ii) the thread variable elimination technique for the generation of irrelevance claims by analyzing C-ordered search graphs of the computation of the goal class, (iii) the reduction methods for unconditional and conditional irrelevance claims.

Our theory was empirically verified with a prototype vocabulary bias shifter constructed with the algorithms developed. It was tried on the kinship example, Gridworld and its variants, and other reachability computations including the generation of Thevenin equivalents of analog circuits (Subramanian, 1988). An interesting empirical fact is that the irrelevance claims used to generate the `max_ancestor` relation have the same form as the ones used to derive the concept of a Thevenin equivalent of a circuit from Kirchoff's and Ohm's laws. Historically, the concept of Thevenin equivalents was discovered almost 100 years after Kirchoff's laws became known. This shows that the framework explored in this paper has the potential to unify vocabulary bias shifts in different domains. The ability to generate shifts from such basic considerations can have significant impact not only on AI, but also in improving our current stock of scientific concepts and formulations.

Appendix I: Derivation of the cost of Horn theories over problem distributions

This is a short summary of the theory of deriving cost estimates for Horn theories on parametric problem distributions, developed in (Subramanian & Hunter, 1992). We illustrate the key concepts of this theory in the context of theoretically estimating time and space costs of the formulation F_1 (introduced in Section 2) over any distribution of `samefamily( $X^b, Y^b$ )` goals. We repeat the description of F_1 here for the reader's convenience, and label the rules for easy reference.

```

s1.  samefamily(X,X).
s2.  samefamily(X,Y) :- ancestor(X,Y).
s3.  samefamily(X,Y) :- ancestor(Y,X).
s4.  samefamily(X,Y) :- ancestor(Z,X), ancestor(Z,Y).
a1.  ancestor(X,Y) :- parent(X,Y).
a2.  ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).

```

Let $C(sf_{bb}, D)$ denote the average cost of solving a `samefamily`(X^b, Y^b) goal drawn from the distribution D . We expect the cost to be a function of the distribution of the bindings for the arguments of `samefamily` goals in D , as well as the topology of the `parent` relation. Note that the goal class D is described parametrically with the name of the goal relation, and the binding status of its arguments. The actual distribution of goals, viz., the specific (X, Y) pairs, is unknown.

Let i be the cost of reducing a goal to its subgoals by a single inference step and let d be the cost of looking up a ground fact. These costs are expressed in the same units (say, dollars), so we can add them directly in cost expressions. Since the goal has a disjunctive definition, it can be solved by any one of the four rules. Our problem solver tries the four rules in the order of entry. Let p_{xx} stand for the probability that a goal drawn from D has the form `samefamily`(X, X); such a goal is solved by the rule `s1`. If the goal is solved by `s1`, there is a total cost of i units incurred. Let p_{xy} stand for the probability that a goal from D is solved by finding an ancestry chain from X to Y , and let $C(A_{bb})$ stand for the average cost of solving an `ancestor` subgoal with both its arguments bound. If a goal from D is successfully solved by an application of `s2` (after a failed attempt using `s1`), followed by a solution of the `ancestor` subgoal, the total cost is $i + i + C(A_{bb})$. Similarly, we let p_{yx} stand for the probability that a goal from D is successfully solved by rule `s3` followed by solution of the resulting `ancestor` subgoal; the associated cost is $[i] + [i + C(A_{bb})] + i + C(A_{bb})$ where the first two terms represent the cost of failed attempts via rules `s1` and `s2` respectively. The last rule `s4` is tried when rules `s1` through `s3` do not yield a solution to the goal. The probability that rule `s4` is used is $1 - p_{xx} - p_{xy} - p_{yx}$; we now calculate the the cost associated with this option. The cost includes that of failure with rules `s1`, `s2` and `s3` — $[i] + [i + C(A_{bb})] + [i + C(A_{bb})]$. With rule `s4`, two conjunctive subgoals are generated. Our solver solves the conjuncts in order from left to right. For every solution to the first conjunct `ancestor`(Z^f, X^b), it solves the second conjunct `ancestor`(Z^b, Y^b) with the binding for Z discovered in the solution of the first conjunct. If the second conjunct fails, then a new solution to the first conjunct is found and the process iterates until a binding Z is found that satisfies both conjuncts (success) or it runs out of solutions to the first conjunct (failure). To calculate average cost for solving this conjunction we need to estimate the average number of solutions found to the first conjunct in the course of solving goals drawn from D . This number, which is the average number of possible ancestors Z found for a given person X , is the average length of a chain in the `parent` relation (denoted as c). Since the four disjuncts defining `samefamily` are mutually exclusive and collectively exhaustive, the expected cost of solving `samefamily`(X^b, Y^b) using F_1 is:

$$C(sf_{bb}) = p_{xx}[i] + p_{xy}[2i + C(A_{bb})] + p_{yx}[3i + 2C(A_{bb})] +$$

$$(1 - p_{xx} - p_{xy} - p_{yx})[4i + 2C(A_{bb}) + c[C(A_{fb}) + C(A_{bb})]]$$

where $C(A_{fb})$ stands for the cost of solving an `ancestor` subgoal with its first argument free and its second argument bound. The above expression is derived automatically by our cost analyzer using the rules in F_1 and information about the control strategy C used by the solver (Subramanian & Hunter, 1992).

Now the analyzer derives cost equations for $C(A_{bb})$ and $C(A_{fb})$ using the rules defining the `ancestor` predicate. We show the steps involved in the derivation of $C(A_{bb})$, the calculation of $C(A_{fb})$ is described next. Rule `a1` is tried first in the attempt to solve an `ancestor`(X^b, Y^b) goal. The probability that a corresponding `parent` fact can be found is denoted $p_{F_{bb}}$ and the cost of solving the goal via `a1` is $i + d$. If rule `a1` fails to yield a solution to the goal (with probability $1 - p_{F_{bb}}$), then rule `a2` is used and the conjuncts `parent`(X^b, Z^f) and `ancestor`(Z^b, Y^b) are solved in order. The cost associated with this second option consists of the cost of failing via rule `a1` ($i + d$) and the average cost of solving the conjunction `parent`(X^b, Z^f) and `ancestor`(Z^b, Y^b). The number of solutions to the first conjunct, on average, is the average number of children ($k \geq 1$) per parent in the `parent` relation.¹³ The cost equation is then formulated as:

$$C(A_{bb}) = p_{F_{bb}}[i + d] + (1 - p_{F_{bb}})[2i + d + k[d + C(A_{bb})]]$$

Our cost analyzer recognizes that this equation is untenable, since the term $C(A_{bb})$ occurs in both the RHS and the LHS of the equation. This pattern occurs because `a2` is a recursive rule, and it signals the need for a cost *recurrence*. We need an induction basis for formulating the recurrence. In structural recursive theories, where computation occurs over lists or over the natural numbers, the basis of the recurrence can be automatically determined by the cost analyzer. However, the theory F_1 is a non-structural theory, where the basis of the recursion is not the size of the input expression. The cost analyzer relies on the user in this case to restructure the cost equations by explicitly identifying the recurrence basis.

We structure the `ancestor`(X^b, Y^b) computation by using the intermediate ancestors between X and Y as the basis of the recurrence. Let $C(A_{bb}, N)$ denote the expected cost of finding a single solution to an `ancestor` goal with N intermediate ancestors. The base case of the computation occurs with the reduction to `parent`(X^b, Y^b) by `a1` and a check to see if the parent fact occurs in the database. The recursive case with N intermediate ancestors consists of reduction of the goal by the second rule into two conjunctive subgoals. Note the relationship of the manually restructured equations to the mechanically generated equation above.

$$C(A_{bb}, 0) = i + d$$

$$C(A_{bb}, N) = (i + d) + i + k[d + C(A_{bb}, N - 1)], N > 1$$

This recurrence is automatically solved to yield:

$$\begin{aligned} C(A_{bb}, N) &= (2i + (k + 1)d) \sum_{j=0}^{N-1} k^j + k^N(i + d), N \geq 1 \\ &= (2i + (k + 1)d) \frac{k^N - 1}{k - 1} + k^N(i + d) \end{aligned}$$

For the goal class $\text{ancestor}(X^f, Y^b)$, unlike the case where both arguments are bound, there are multiple solutions for X . In (Subramanian & Hunter, 1992), we describe how the number of solutions of a recursive predicate are calculated. The cost of obtaining the first solution is $i + d$ since the goal is solved via `a1`. For subsequent solutions, the cost $C(A_{fb})$ per solution derived by the analyzer using `a2` is:

$$C(A_{fb}) = d + C(A_{bb})$$

Substituting the various terms into the average cost of solving a goal drawn from D , the system derives:

$$\begin{aligned} C(sf_{bb}) = & p_{xx}[i] + p_{xy}[2i + O(k^c)] + p_{yx}[3i + O(k^c)] + \\ & (1 - p_{xx} - p_{xy} - p_{yx})[4i + O(k^c) + d + c[O(k^c)]] \end{aligned}$$

The average cost of a goal in D with the formulation F_1 is $O(k^c)$. Thus it is exponential in the local branching factor of `parent`. While the parameters p_{xx} , p_{xy} and p_{yx} model the distribution D , the parameters c and k are characteristic of the `parent` relation. These are the only aspects of the distribution and the formulation that are relevant to the cost.

We model static space costs only and not the dynamic space cost (cost of run-time stacks). This is because, for the kinds of computations considered in this paper, the dynamic space cost is reflected in the time cost itself. Let s denote the cost of storing a fact or a rule in the database. The size of a ground relation r is denoted as $\text{size}(r)$. The space cost of F_1 is largely determined by the size of the `parent` relation.

$$\text{Space_Cost}(F_1, D) = \text{size}(\text{parent})$$

Appendix II: Constructing C-ordered search graphs

In this section, we describe how to construct C-ordered search graphs for definite clause Horn theories and definite goals. These graphs are used by the irrelevance analyzer of formulations. For expositional ease, we begin the construction of these graphs for ground goals. We then describe how goal classes can be handled, and show that the complexity of construction is polynomial in the number of rules in a theory and exponential in the arities of the defined relations. For theories with defined relations of bounded arities, the construction procedure is polynomial in the size of the theory.

Since we work with definite clause Horn theories T , we use SLD resolution (SLDNF resolution is used for safe, stratified Horn theories with negation) to prove goals drawn from the problem distribution.

Definition 11 (Lloyd) *We will say that G' is derived from G and the definite clause $H : -A_1, \dots, A_n$ by SLD resolution if G is the goal $G_1, \dots, G_i, \dots, G_k$, σ is the most general unifier of H and G_i , and G' is $(G_1, \dots, G_{i-1}, A_1, \dots, A_n, G_{i+1}, G_k) \cdot \sigma$, where $\cdot$ is the operator that applies a substitution to a formula.*

Consider the simple definite clause theory about kinship defined below. We will use this theory to illustrate our definitions.

```

a1  ancestor(X,Y) :- parent(X,Y).
a2  ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).

```

Consider the ground goal $\text{ancestor}(a,b)$. We can use SLD resolution with rule a1 to obtain the new subgoal $\text{parent}(a,b)$ with most general unifier $\{X/a, Y/b\}$. SLD resolution with rule a2 yields the new conjunction $\text{parent}(a,Z), \text{ancestor}(Z,b)$ with most general unifier $\{X/a, Y/b\}$. We now provide a constructive definition of a C-ordered search graph for G in (T, C) . C is taken to be the default policy of processing literals from left to right and considering rules in the order in which they occur in the theory.

Definition 12 Let T be a set of definite clauses and G a definite goal. A C-ordered search graph for T in (G, C) is the rooted, directed graph defined below. Each node of the graph contains a possibly empty list of definite goals. The depth of a node n in the graph is the length of the shortest path from the root to n .

1. (Root node): The root node is the list containing G .
2. (Cycle detection): Let $A_1, \dots, A_k$ ($k \geq 1$) be a node in the tree. If there exists another node in the tree with the literal list $B_1, \dots, B_k$ ($k \geq 1$) where the A list is identical to the B list up to variable renaming, there is a directed unification edge (doubly-lined back edges in our figures) from the node at the greater depth to the node at the smaller depth.
3. (Goal reduction): Let $A_1, \dots, A_k$ ($k \geq 1$) be a node in the tree. For the atom A_1 , let $R = \{A :- B_1, \dots, B_q \in T \mid \exists \sigma \text{ } A_1 \text{ and } A \text{ are unifiable with mgu } \sigma\}$. This node has $|R|$ children: each child is a node of the form $(B_1, \dots, B_q, A_2, \dots, A_k) \cdot \sigma$. The edge from the node to each child is labeled by the name of the rule in R and the substitution σ used in its generation. If A_1 is an arithmetic computation or a built-in predicate (e.g., equality), its child nodes are of the form $\{A_2, \dots, A_k\} \cdot \sigma$ where σ consists of the substitutions obtained by solving A_1 . The edge connecting the parent and child node in this case is labeled a .

Nodes which have the empty clause have no children. Nodes for which no rule head in T unifies with the first literal also have no children. Because of the cycle detection step in the construction, the C-ordered search-graph for a theory T and a goal scheme G is a finite structure. Each branch of the C-ordered search tree is an attempt to derive G in T using C . There are two types of branches in this tree: successful branches corresponding to successful derivations of G (these end in the empty clause), and failed branches with non-empty leaves (Lloyd, 1987). The C-ordered search graph can be viewed as the set of possible schematic proofs of G in T .

Definition 13 A proof of G in (T, C) is a path in the C-ordered search graph of G in T that starts at the root and terminates in a leaf node with the empty clause.

We now describe how goal classes are represented in our framework. Let us consider the goal class $\text{ancestor}(X^b, Y^b)$, where both arguments X and Y are bound. The

actual constants will be provided at run-time. Our scheme for defining goal classes by superscripting variables, with b or f standing for whether they are bound or free, follows the convention in the database optimization literature (Ullman 1989). The definitions below show how annotations on variables in goals and subgoals are propagated by an SLD resolution step.

Definition 14 *An annotated literal $G(X^{a_1})$ unifies with the annotated literal*

$G(P^{a_2})$ where X and P are annotated variables, when (i) $G(X)$ and $G(P)$ unify by the classical rules, and (ii) annotations $a_1, a_2 \in \{b, f\}$ are compatible. The pairs (b, f) , (f, b) and (f, f) are compatible. If a variable with annotation b is unified with a variable with annotation f , then both variables are coerced to have annotation b . When two variables with annotations f are unified, they continue to have the annotation f .

This definition can be naturally extended to cover unification of literals with more than one variable. We require that the annotations of unified variables be pairwise compatible.

Definition 15 *We will say that the annotated set of goal literals G' is derived from the annotated goal literal G and the definite clause $H : -A_1, \dots, A_l$ by SLD resolution if σ is the most general unifier of H (with every variable in H annotated by f) and G computed according to the rules in Definition 14, and G' is $(B_1, \dots, B_l) \cdot \sigma$, where $\cdot$ is the operator that applies a substitution to a formula and each B_i is A_i with its variables annotated according to the following rules¹⁴:*

1. *If variable X in A_i occurs in the head H and occurs in no A_j for $j < i$, then X inherits the annotation of X in H .*
2. *If variable X in A_i does not occur in H and occurs in no A_j for $j < i$, then it obtains the annotation f .*
3. *If variable X in A_i occurs in some A_j for $j < i$, it obtains the annotation b .*

SLD resolution of the goal class $\text{ancestor}(X^b, Y^b)$ with rule a_1 yields the subgoal class $\text{parent}(X^b, Y^b)$ (case 1 of the above definition). SLD resolution of $\text{ancestor}(X^b, Y^b)$ with rule a_2 produces the conjunctive subgoal class $\text{parent}(X^b, Z^f), \text{ancestor}(Z^b, Y^b)$. The variable Z in the parent literal obtains the annotation f because it is the first occurrence of a variable that appears only in the body of a_2 (case 2 of the above definition). In the following ancestor literal, Z is classified as a bound variable because it is instantiated by the previous literal (case 3 of the definition above).

Using Definition 15, we can generate schematic C-ordered search graphs for goal classes instead of for specific ground goals. A statically generated C-ordered search graph is shown in Figure 3. Nodes that are alphabetic variants of each other are connected by double edges. We can easily prove the following from Definition 12.

Proposition 8 *The construction of a C-ordered search graph for definite goal class G and definite clause theory T is polynomial in the number of rules in T and exponential in the arity of the defined predicates in T .*

Appendix III: Verifying irrelevance claims

From Definition 8, we can derive the following lemmas that can be used for proving that a given formula f is irrelevant to a goal scheme g . Recall that all goal schemes are positive literals or conjunctions of positive literals. We decompose non-atomic formulae f and g by these lemmas, and use the algorithm in Table 4 for verification of irrelevance claims involving atomic g 's and f 's.

1. $irrelevant(f \wedge h, g, (T, C)) \equiv irrelevant(f, g, (T, C)) \vee irrelevant(h, g, (T, C))$
where $f \wedge h$ is consistent.
2. $\neg irrelevant(\neg f, g, (T, C)) \Rightarrow irrelevant(f, g, (T, C))$ if $T \vdash_C g$ and C
has negation by failure.
3. $irrelevant(f, g_1 \wedge g_2, (T, C)) \Rightarrow irrelevant(f, g_1, (T, C)) \wedge$
 $irrelevant(f, g_2, (T, C)).$
4. $irrelevant(f, g_1, (T, C)) \vee irrelevant(f, g_2, (T, C)) \Rightarrow$
 $irrelevant(f, (g_1 \vee g_2), (T, C))$

We have derived more lemmas from Definition 8 (Subramanian, 1988); the above constitute a representative sample. Our implementation of the irrelevance claim verifier in C-Prolog breaks down non-atomic goal and fact schemes by these irrelevance lemmas and uses the Procedure in Table 4 to evaluate the truth of claims involving atomic facts and goals.

Algorithm *verify* in Table 4 performs a complete depth-first traversal of S , the C-ordered search graph for g in (T, C) , and computes the set P of successful paths in S . The C-ordered search graph of a goal scheme G in the theory T is a representation of the search space of proofs of g in (T, C) . It identifies elements of T that are relevant to the goal, as well as opportunities for improving time and space costs in the computation. Each node in the graph is a set of subgoals ordered by C . The variables in subgoals are annotated to indicate whether they are bound or free. This annotation is done automatically by an algorithm that generates the graph from (T, C) and g (see Appendix II). Each single arc denotes a one-step inference using a rule in T . A proof of G in (T, C) is a path in the C-ordered search graph that starts at the root and terminates in a leaf node with the empty clause. Note that in Figure 3, there are several paths from the root that end in the empty clause. Since the underlying theory is recursive, there are doubly-lined back edges in the graph linking sets of subgoals that are identical up to variable renaming. Note that in the construction of the C-ordered search graph S , we do not expand literals corresponding to primitive relations. Cycles (doubly-lined back edges in S) are traversed exactly once. This decision reduces the time complexity of construction of P to $O(size(S))$ where $size(S)$ is the sum of the number of edges and vertices in S . We associate an occur-flag with each path in P . The occur-flag for $p \in P$ is set if p contains a node that has a literal

that unifies with f . We also associate with every path p in P , a set $C(p)$ of literals, that are solved by looking up primitive relations (e.g. parent literals in our running example), or by solving arithmetic or unification constraints.

In this paper, we are concerned with single answers to goals. We distinguish between goals with variables annotated as bound and those with variables annotated as free. To see why, consider the theory $T = \{g(X) : -f(X), g(X) : -h(X), f(a), h(b)\}$.¹⁵ If the goal formula is $g(X^f)$, we are interested in a single answer to the $g(X)$ goal where X is free at the start of the computation. For this goal class, we can deem f irrelevant by checking if there is some path p in P that does not have its occur-flag with respect to f set. Then, provided $C(p)$ holds, f is irrelevant to g . Pruning the derivation of g via f is legitimate because we are interested in any one solution to $g(X^f)$, and not all solutions. If, however, the goal class is $g(X^b)$, we are interested in showing that g holds of some constant specified externally at the start of the computation. We now need a more sophisticated scheme for verifying that f does not contribute to any specific solution of g . In the previous example f is irrelevant to $g(b)$, but not irrelevant to $g(a)$. The path in S involving f is the only path that yields the solution $g(a)$. So, provided $X \neq a$, f is irrelevant to $g(X^b)$. This analysis requires that (1) we compute all solutions to g via the paths in P on which f lies, and (2) we verify that it contains substitutions that are not generated by other paths in P . This amounts to computing all solutions for g — a process which may not terminate (e.g., for left-recursive theories under our control policy). If all solutions for g can be generated finitely, then the procedure *verify* is complete for definite Horn theories.

Proposition 9 *Algorithm $\text{verify}(f, g, (T, C))$ is sound for definite Horn theories T , for atomic g 's and f 's with all variables in f annotated as free.*

Proposition 10 *Algorithm $\text{verify}(f, g, (T, C))$ is complete for definite Horn theories T for which all solutions to g can be generated finitely.*

$\text{verify}(f, g, (T, C))$ is not complete for non-Horn theories. This follows trivially from the fact that our $\vdash_C$ procedure is not complete for non-Horn theories. Consider $T = \{f \vee g, g : -f\}$. The $\vdash_C$ procedure cannot establish g from these two clauses. So the search graph S cannot be constructed and therefore *verify* cannot infer the relevance or irrelevance of f with respect to g .

Proposition 11 *Algorithm $\text{verify}(f, g, (T, C))$ has worst-case time complexity linear in the size of S for (1) definite Horn theories without recursion and (2) definite Horn theories with recursion provided all variables in g are annotated as free. For definite Horn theories with recursion, and with variables in g annotated as bound, the worst-case time complexity is bounded by the number of possible instantiations of S using the primitive relations.*

Notes

1. This example is due to Jerry Hobbs (Hobbs, 1985).

2. Definite goals are formulae consisting of a single positive literal or a conjunction of positive literals.
3. In this paper, we model lookup costs as constants assuming indexing on all arguments. In practice, a hash table organization can give us lookup times that are a function of the log of the number of entries. The results of the paper are not sensitive to this difference. We could just as easily have modeled the lookup cost with logs.
4. This is because our results depend on the existence of a unique Herbrand model for Horn theories.
5. We omit function symbols here for expositional clarity.
6. Since we do not consider function symbols here, we are restricted in the type of object mappings we can consider. The general mapping is from constants in T_2 to terms in the vocabulary of T_1 .
7. $\text{argmax}_x F(x)$ denotes an x that maximizes the function $F(x)$.
8. There are many classes of explanations that can be constructed by considering other meta-level properties of a vocabulary item. An example is a symmetry explanation. Such an explanation captures the fact that two vocabulary items have symmetric roles in a computation. See (Ellman, 1993) for the use of such explanations.
9. Technically, we treat $\text{parent}(X, Y)$ as a term standing for the parent relation, so that our generator is a first-order system. Our generator is implemented in C-Prolog and it uses SLDNF resolution.
10. Let one_step be a binary relation on a set of states that specifies all one-step moves in the search space. The transitive closure reachable can be formulated as the rule below. Note that it is identical structurally to the transitivity rule of ancestry.

$$\text{reachable}(\text{Start}, \text{End}) \text{ :- } \text{one_step}(\text{Start}, \text{Int}), \text{reachable}(\text{Int}, \text{End}).$$

11. The formulation evaluator's sample complexity is polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ (Subramanian & Hunter, 1992).
12. Even though the reduction procedure takes subsets of T , we are allowed to augment T by propositions in its deductive closure. The conditional irrelevance claims generate reductions that are subsets of the deductive closure of T .
13. For each primitive relation, the cost analyzer computes the average number of solutions for each binding category (for binary relations, there are four categories bb , bf , fb and ff).
14. Note that H acquires new annotations after being unified with G .
15. I am indebted to an anonymous reviewer for this example.

References

- Amarel, S. (1968). On representation of problems of reasoning about actions. In Donald Michie, editor, *Machine Intelligence 3*, pages 131–171. Edinburgh University Press, Edinburgh, Scotland.
- Buntine, W. (1990). Myths and legends in learning classification rules. In *Proceedings of AAAI-90*. Morgan Kaufmann.
- Cohen, W. (1990). Learning approximate control rules of high utility. In *Proceedings of ML90*, pages 268–276.
- Ellman, T. (1993). Abstraction by approximate symmetry. In *Proceedings of IJCAI-93*, Chambéry, France.
- Fu, L.-M. & Buchanan, B.G. (1985). Learning intermediate knowledge in constructing a hierarchical knowledge base. In *Proc. of IJCAI-85*, pages 659–666, Los Angeles, CA, August 1985. IJCAI, Inc.
- Flann, N. (1988). Improving problem solving performance by example-guided reformulation of knowledge. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York. Philips Laboratories.
- Gordon, D. (1990). *Active bias adjustment for incremental supervised concept learning*. PhD thesis, University of Maryland.
- Haussler, D. (1990). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. In J. Shavlik & T. Dietterich, editors, *Readings in machine learning*, pages 205–227. Morgan Kaufmann.
- Hobbs, J. (1985). Granularity. In *Proceedings of IJCAI-85*. Morgan Kaufmann.
- Keller, R.M. (1987). *Learning in Context*. PhD thesis, Rutgers University.

- Keller, R., Mitchell, T., & Kedar-Cabelli, S. (1986). Explanation-based learning: A unified view. *Machine Learning*, 1(1):47–80.
- Korf, R.E. (1980). Toward a model of representation changes. *Artificial Intelligence*, 14(1):41–78.
- Korf, R.E. (1987). Planning as search: a quantitative approach. *Artificial Intelligence*, 33(1):65–88.
- Knoblock, C.A., Tenenber, J.D., & Yang, Q. (1991). Characterizing abstraction hierarchies for planning. In *Proceedings of AAAI-91*.
- Lloyd, J.W. (1987). *Foundations of Logic Programming (second extended edition)*. Springer-Verlag.
- Laird, J.E., Newell, A., & Rosenbloom, P.S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64.
- Lowry, M. (1988). Problem reformulation and ADT. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York. Philips Laboratories.
- Mitchell, T. (1990). The need for biases in learning generalizations. In J. Shavlik & T. Dietterich, editors, *Readings in Machine Learning*, pages 184–191. Morgan Kaufmann.
- Mitchell, T., Utgoff, P.E. & Banerji, R.B. (1982). Learning by experimentation: Acquiring and refining problem-solving heuristics. In *Machine Learning*. Tioga.
- Muggleton, S. (1988). Constructive induction in first-order logic. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York. Philips Laboratories.
- Provost, F. (1992). *Policies for the selection of inductive bias in machine learning*. PhD thesis, University of Pittsburgh.
- Russell, S. & Grosz, B. (1987). A declarative approach to bias in concept learning. In *Proceedings of AAAI-87*, pages 505–510. Morgan Kaufmann.
- Riddle, P. (1988). An approach to learning problem decomposition schemas and iterative macro-operators. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York. Philips Laboratories.
- Rosenbloom, P. (1983). *The chunking of goal hierarchies: a model of practice and stimulus-response capability*. PhD thesis, Carnegie-Mellon University.
- Schlimmer, J. & Proca, A. (1988). Learning, problem-solving and representation change. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York. Philips Laboratories.
- Shmueli, O. (1987). Decidability and expressiveness aspects of logic queries. In *Proc. 6th ACM PODS*, pages 237–249.
- Spears, W. & Gordon, D. (1992). Is consistency harmful? In *Proc. of the Workshop on biases in inductive learning, ML-92*.
- Subramanian, D. & Genesereth, M.R. (1987). The relevance of irrelevance. In *Proceedings of IJCAI-87*, Milan, Italy.
- Subramanian, D. & Hunter, S. (1992). Measuring utility and the design of provably good EBL algorithms. In *Proceedings of ML-92*.
- Subramanian, D. (1988). *A Theory of Justified Reformulations*. PhD thesis, Stanford University.
- Tambe, M. & Rosenbloom, P. (1989). Eliminating expensive chunks by restricting expressiveness. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 731–737. Morgan Kaufmann.
- Ullman, J.D. (1989). *Principles of database and knowledge-based systems, Vol II*. Computer Science Press.
- Utgoff, P.E. (1986). *Shift of Bias for Inductive Concept Learning*. PhD thesis, Rutgers University.
- Webber, A. (1993). *Optimizing functional programs by reasoning about irrelevant computation*. PhD thesis, Cornell University.

Received December 21, 1993

Final Manuscript September 6, 1994