Shifting vocabulary bias in speedup learning

Devika Subramanian

April 18, 1994

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.

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 [Mit90] 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 [Utg86, Hau90, Bun90, RG87, Pro92, Gor90, GS92], biases in speedup learning have received little attention.

Speedup learning algorithms improve performance of solvers over given distributions of problems. Speedup learning models human learning from practice or experience [Ros83]. The objective of a speedup learner is to find good tradeoffs between coverage of the learned solvers over the given distribution and time and space cost. 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.

Bias plays a key role in the design of speedup learners. Building on Mitchell's, Utgoff's, as well as 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 goal distribution". We do not limit ourselves to solvers that only speed up observed training instances, thus treating instances as additional sources of bias, as suggested in [GS92].

In this paper, we consider a family of speedup learning methods that improve problem solving in theories formulated as definite Horn clauses (resp. safe, stratified Horn clauses). SLD resolution (resp. SLDNF resolution) [Llo87] is a complete inference method for such theories. 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, whereas 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 (T, C) 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. This space is generally not modified during learning and performance guarantees on new solvers are made relative to this space. Most speedup learning methods limit themselves to modifications that preserve the ontology of T and C. 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? In this paper we consider a multi-dimensional evaluation basis: coverage, time and space efficiency. We then experiment with various policies for trading off these attributes against each other. This is done by specifying objective functions that explore the space of specific compromises between coverage, and time and space costs.

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. It 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 are determined by a principle of computational economy [SG87]. Explicit minimization of wasted computational effort can both justify and generate an interesting class of vocabulary shifts. We call this the irrelevance principle. The shifts sanctioned by the 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 it[1], and we move easily between instant and interval-based representations of time. We show the following in this paper.

[1]This example is due to Jerry Hobbs [Hob85].

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 *explains* the generation of coarser theories with better computational properties. We demonstrate how explanations of irrelevant computation can be used to *generate* more efficient theories adapted for specific computational classes.

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 [TR89]. 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 definite Horn theory F_1 interpreted by a procedurally encoded C_o — SLD resolution, with rules are chosen in order of entry, and conjuncts solved from left to right. SLD-resolution is a sound and complete inference procedure for definite Horn theories on definite goals which are goal formula consisting of a single or a conjunction of positive literals.

```
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 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`[2]. 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 $O(N)$.

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

$$\texttt{max\_ancestor(X,Y)} \equiv \texttt{ancestor(X,Y)} \;\wedge\; \neg\, \texttt{parent(Z,X)}$$

(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[3]. 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[4]. 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) [5]. 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.

We have presented three different solvers for the kinship problem. Each solver had the same procedurally encoded interpretation policy, C_o. However, the associated theories were characterized by different ontological choices. The vocabulary commitments of theories F_1, F_2 and F_3 are shown in the table below. Relations or predicates are annotated by their type: primitive, computed or hybrid. A hybrid relation is part primitive and part computed. $\mathcal{P} = Dom(\texttt{parent}) \cup Range(\texttt{parent})$ stands for the set of people in the database.

Theory	Objects	Primitive Relations	Computed Relations
F_1	$\mathcal{P}$	`parent`	`ancestor`,`samefamily`
F_2	$\mathcal{P}$	`max_ancestor`	`samefamily`
F_3	$\mathcal{P}$	`samefamily`	

[2] This condition is required for proving that the computation terminates.

[3] Z is a variable that occurs only in the body of the rule and not in the head.

[4] 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 have easily modeled the lookup cost with logs.

[5] From a practical standpoint, constructing this formulation makes sense in a world where the one-step inference that reduces the `samefamily` goal to the `ancestor` goals is much more expensive than a direct lookup of `samefamily` and the cost of storing the entire `samefamily` relation.

We can now pose the following questions.

1. How do we evaluate vocabulary choices with respect to a distribution? What is the "best" choice for the kinship problem for a given control policy C and a given problem distribution?

2. What is the space of shifts of vocabulary bias? How can we systematically generate them? What principles govern shifts in this space? What algorithmic processes can identify the "best" vocabulary bias in this space?

These two questions are addressed in detail in the next two sections. We now 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 also 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

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[6]. 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 $v_1 \geq_D v_2$, if $c_1 \succeq c_2$ and $t_1 \succeq t_2$ and $s_1 \succeq s_2$ and at least one of the inequalities is strict. Solver (T_1, C_1) dominates solver (T_2, C_2) if

[6] Previous work on speedup learning in the context of one-level solvers evaluates them along the dimensions of coverage and average time cost. [Coh90] is an example. We add space cost in our evaluation because it allows us to discriminate and evaluate a (T, C) pair which is essentially a lookup table of answers to the given goal distribution D.

$(c_1, t_1, s_1) \geq_D (c_2, t_2, s_2)$. 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 [SH92].

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[7] of the parameters; for instance we can represent coverage as two numbers: one representing the probability that the solver will provide no 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. A detailed study relating the nature of the value function to characteristics of the problem domain is outside the scope of this paper.

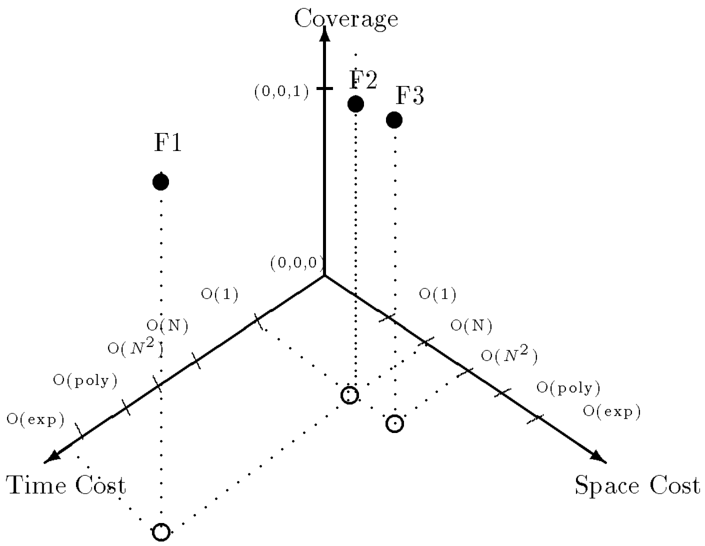

Figure 1: Plotting the solvers in the kinship problem on the three dimensional evaluation space. The projections on the time and space cost axes are sufficient to compare these solvers, since they all have a coverage of 1.0. Preferred directions are toward the origin for time and space cost and toward 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 recog-

[7] I am indebted to an anonymous reviewer for this suggestion.

nizability of global optima. For instance, is (F_2, C_o) the globally optimal solver for the distribution D of `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 decrease 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. We can keep trading loss of coverage for gain in space cost till the difference in these two marginal costs becomes zero. Such a point 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 isn't such a direct relationship between the three value dimensions. Thus there is no efficient method for verification of the global optimum short of examining the entire space. Therefore, we are satisfied with formulations that are local optima of the value function v in the space of solvers. To decide that we are at a local optimum, we need to show that no one-step movement improves v. We therefore need to describe how the learner systematically generates points in the space of solvers.

3.2 Generating the space of solvers

Each point (T, C) in the space of solvers embodies a specific ontological commitment reflected in its choice 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.[8]

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.[9] 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[10] from the old vocabulary, but the old one cannot be reconstructed from the new vocabulary. Thus distinctions are lost in the vocabulary shift. 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 from the domain of Euclidean geometry proofs involves the introduction of a new object (a perpendicular bisector) to make it easier to solve the problem of proving that the base angles of an isoceles 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

[8] 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.

[9] We omit function symbols here for expositional clarity.

[10] We make this notion very precise in this section.

important and useful class that includes abstractions and (2) we have systematic ways of defining the space of these shifts and domain-independent principles of traversing the space.

Before we study granularity-coarsening shifts, we define the class of definite Horn theories. All rules in a definite Horn theory are of the form $A \Leftarrow B_1, \ldots B_m$ where the head A is a positive literal and the B_i's are also positive literals. We adopt this restriction to make exposition simpler. Our results also hold for Horn theories with negated literals in bodies of the rules, as long as the rules are safe and stratified [Ull89]. 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[11] from p to q. 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 below holds with the word *definite* being replaced by *safe and stratified* and *SLD* replaced by *SLDNF* resolution.

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 $f_o : c_2 \rightarrow \mathcal{P}(c_1)$ which maps objects constants in T_2 to subsets of object constants in T_1. $\mathcal{P}(x)$ 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 $r(\vec{X}) \equiv p_1(Y_1), \ldots p_m(Y_m)$, 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.*

A granularity coarsening shift between vocabularies is a forgetful map: it is an information-losing transformation. We make this precise when we discuss models of a first order theory T. 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 $a \in Vocab(T_1), Vocab(T_2)$[12]. In essence, f_o allows us to carve up the space 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., $\mathtt{f_r}(\mathtt{max\_ancestor(X,Y)}) \equiv \mathtt{ancestor(X,Y)} \ \wedge \ \neg\, \mathtt{parent(Z,X)}$.

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

We will say that $Vocab(T_2)$ 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 $g : Vocab(T_1) \rightarrow$

[11] A dependency graph has predicate names as nodes, and an arc is drawn from p to q if there is a rule with p at its head and q (in positive or negative form) in its body of the rule set from p to q.

[12] 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.

$Vocab(T_2)$. $Vocab(T_2)$ 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 `parent(X,Y)` and `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*

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

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. This is essential for evaluating coverage over a goal distribution. (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 $f : V \to Vocab(T)$, 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 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 [Llo87]. The least Herbrand model is the intersection of all Herbrand models of the theory. The mapping f between vocabularies of two theories induces a map between the least Herbrand models of those theories.

The Herbrand universe U of a definite Horn theory 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. Each term in U_2 stands for a collection of terms in U_1.

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

In our kinship example, the Herbrand universes of the F_i's are all the same because f maps each object constant to itself.

The Herbrand base B of a definite theory is the set of all ground positive literals that can be formed by using predicate symbols from T and ground terms from the Herbrand universe U. A Herbrand model of a definite theory T is a subset of the Herbrand base. Given a formula ϕ with free variables $v_1, \ldots v_k$ and Herbrand model M_1 over a Herbrand universe U_1, we say that $\models_{U_1} \phi[[a_1, \ldots, a_k]]$ where v_i is assigned to $a_i \in U_1$ if $\phi[[a_1, \ldots, a_k]]$ is true in M_1. With this formula ϕ we can associate the k-ary relation $\{< a_1, \ldots, a_k >: \models_{U_1} \phi[[a_1, \ldots, a_k]]\}$ 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

$$M_2 = \{p(b_1, \ldots, b_k) \in U_2 \mid f(p) = \phi, \models_{U_1} \phi[[a_1, \ldots, a_k]], a_i \in f(b_i)\}$$

Proposition 2 *The least Herbrand model (LHM) of T_2 can be 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 3 *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 f to the space of 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 Proposition 2 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.

$$LHM(T_2) = Construct(LHM(T_1), f) \text{ using Proposition 2.} \tag{1}$$

We take a simple propositional example to illustrate how this equation can be used for the construction of an encoding of T_2. 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 $\{f, f \Rightarrow \neg g\}$. Thus, while this equation places constraints on the encoding of T_2, it does not specify it uniquely. In the next subsection we show how we use the bias in the encoding of T_1 to restrict the space of possible reconstructions of T_2. Generating solutions to the reconstruction equation 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. So we reconstruct T_2 by performing modifications on T_1, instead of "from scratch".

Consider the construction of F_2 from F_1 in our kinship example. F_1 consists of the following definite Horn theory, along with ground `parent` propositions, which are not shown below.

```
1.  ancestor(X,Y) :- parent(X,Y).
2.  ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
3.  samefamily(X,X).
4.  samefamily(X,Y) :- ancestor(X,Y).
5.  samefamily(X,Y) :- ancestor(Y,X).
6.  samefamily(X,Y) :- ancestor(Z,X), ancestor(Z,Y).
```

The map $f : Vocab(F_2) \rightarrow Vocab(F_1)$ is a 1-1 map on objects and the following relation map.

$$\begin{aligned} \texttt{f(max\_ancestor(X,Y))} &\equiv \texttt{ancestor(X,Y)} \wedge \neg\,\texttt{parent(Z,X)} \\ \texttt{f(samefamily(X,Y))} &\equiv \texttt{samefamily(X,Y)}. \end{aligned}$$

In F_2, `samefamily` is a computed relation while `max_ancestor` is a primitive relation. The least Herbrand model of F_1 under the closed world assumption consists of the set of given parent pairs, ancestor pairs that form the transitive closure of the parent relation, and the samefamily relation which consists of all (x, y) pairs where both members of the pair belong to the same parent tree. The interpretation of the `max_ancestor` relation symbol is a set of pairs (x, y) where x is the maximal ancestor of the tree to which y belongs. The definition of `max_ancestor` is not a definite Horn clause. This is why we represent it in F_2 as a primitive relation because we wish to have the operations in the formulation space preserve the property of the rule set being definite and Horn. The minimal Herbrand model of F_2 consists of the objects in F_1 together with the extensions of `max_ancestor` and `samefamily`. Given that `max_ancestor` is primitive in F_2, we compute all ground instances of the relation from F_1 and the definitions in f. That is, we find $\{(X, Y) \mid F_1 \cup f \vdash_{C_o} \texttt{max\_ancestor(X,Y)}\}$. Since `samefamily` is computed, we need to define it using other relation symbols in F_2. `max_ancestor` is the only other relation in F_2, so we replace every occurrence of `ancestor` in sentences `4` through `6` of F_1 by `max_ancestor` to generate F_2. Sentences `1` and `2` of F_1 have no counterparts in F_2 because they make distinctions that F_2 does not. The process of constructing F_2 given f and F_1 is non-trivial. We can verify that F_2 composed of the primitive `max_ancestor` relation and the axioms `3`, `4`, `5` and `6` of F_1 with every occurrence of `ancestor` replaced by `max_ancestor` has the models specified by the reconstruction equation 1.

Definition 4 *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. The coverage of T_2 over D is the proportion of goals in D that it can answer.

In our example, F_2 has good computational and coverage characteristics. As shown in Section 2, it has an average time cost of $O(1)$ for our goal distribution and a space cost of $O(N)$.

Definition 5 *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 initial theory T_o and control policy C, if there exist encodings T_1 and T_2 that satisfy the reconstruction equation such that $v(T_2) \succeq_D v(T_1)$, where v is the value function on formulations.*

Given this ordering between granularity coarsening shifts, we can state the goal of a speedup learner precisely[13]: Find

$$argmax_{\substack{V \in VS(Vocab(T_o)) \\ f: V \rightarrow Vocab(T_o)}} f$$

[13] $argmax_x F(x)$ denotes an x that maximizes the function $F(x)$.

This is a specification of a global optimum in the space $VS(Vocab(T_o))$ of vocabularies. We only wish to identify local optima in VS. The definition 3 of $VS(Vocab(T_o))$ is precise but does not exhibit the type of local structure needed to set up a hill-climbing search for a correct and locally optimal vocabulary bias. We therefore define four parametric operators that generate the vocabulary space starting with $Vocab(T_o)$. We also sketch implementations of the vocabulary bias space operators that generate new theories satisfying the reconstruction equation. These are parametric "theory hacks" for implementing vocabulary transformations. The detailed descriptions of these implementations are in the next section.

1. *Adding definable relations.* This transformation preserves information. A definable relation is constructed from the existing set of relations and objects; it is a repackaging of the elements of the formulation. The addition of the primitive relation `max_ancestor` in the construction of formulation F_2 is an instance of the transformation. We denote the operation `add(r,`ϕ`)`, where ϕ is the defining formula for the relation symbol `r`.

   If the rule defining r is a Horn rule whose addition leaves the theory T definite (resp. safe and stratified), then we simply add the rule defining r to T. Otherwise, we add r as a primitive relation by precomputing it.

2. *Deleting relations.* This is an information losing transformation. The elimination of the parent relation in the construction of F_2 is an example. In terms of f, this amounts to making the relation map of f an <u>into</u> mapping. We denote the operation `del(r)` where `r` is a relation symbol.

   To delete the relation `r`, we need to revise the theory so that no `r(`$\vec{x}$`)` can be derived in the theory. The revision employs a notion of minimality. The smallest change (in the sense of set inclusion) to prevent the derivation of a `r(`$\vec{x}$`)` fact is implemented. This is so as to exert as little effort as possible to perform the revision.

   There are two cases to be considered. If `r` is a primitive relation, there are no rules that conclude `r` in the theory. The removal of every ground instance of `r` from the theory is sufficient to delete a primitive `r`. We can also eliminate rules that contain an `r` literal in their antecedent list. To eliminate a derived relation `r`, we need to prevent the inference of `r` using any of the rules in the theory that conclude `r`. We illustrate this with an example. Suppose we wish to eliminate the `ancestor` relation from F_1. There are two rules that derive `ancestor` facts. The first reduces it to the `parent` relation. The second reduces an `ancestor` literal to a conjunction that includes a `parent` literal. Thus eliminating the primitive `parent` relation breaks both inference paths. This is a minimal revision (in the sense of set inclusion) of F_1 that deletes `ancestor`. Another minimal revision consists of the elimination of both rules that derive `ancestor`. Our implemented system generates both revisions and prefers the one that yields a formulation with smaller cost according to the value function v.

3. *Quotienting relations by defining equivalence classes on objects.* This is an 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 have no instances of quotienting in our kinship example. We denote the operation `quot(r,`$\approx$`)`, where `r` is a relation symbol denoting R.

   If `r` is primitive, the quotient set r_{approx} is extensionally calculated. This proceeds in two steps: (1) determination of the canonical elements $[x] = \{y \mid y \approx x, x, y \in \mathtt{r}\}$, where $[x]$ is the chosen representative, (2) calculating the quotient: $r_{approx} = \{[x] \mid x \in \mathtt{r}\}$. To quotient a defined or computed relation `r`, we need to explicitly calculate canonical elements from the $\approx$ relation (a forward chaining computation) and store it in the binary `canonical` relation. The quotient relation can then be defined as $\mathtt{r}_{approx}(X) : -\mathtt{r}(Y), \mathtt{canonical}(X,Y)$..

4. *Converting computed relations to primitive relations and vice versa.* In our problem solver, the cost of solving primitive and computed relations is different enough, that this distinction is important. The construction of F_3 from F_2 involves converting `samefamily` from a computed to a primitive relation. We denote this operation `convert(r,l)` where `l` is the new label for relation symbol `r`.

   To convert a computed relation `r` to a primitive one, we forward chain on the theory, generating all derivable `r` facts, and store them explicitly as ground facts. This technique is used for materializing finite relations only. To convert a primitive relation to a computed one, we need to find a defining formula whose extension is the primitive relation. This is treated as a learning problem where the tuples in the primitive relation are instances, and the definition for the computed form of the relation is acquired by k-CNF or k-DL learning.

Proposition 4 *Every granularity coarsening shift $f : Vocab(T_2) \rightarrow Vocab(T_1)$ can be constructed from the four operators:* `add(r,`ϕ`)`, `del(r)`, `quot(r,`$\approx$`)` *and* `convert(r,l)`.

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

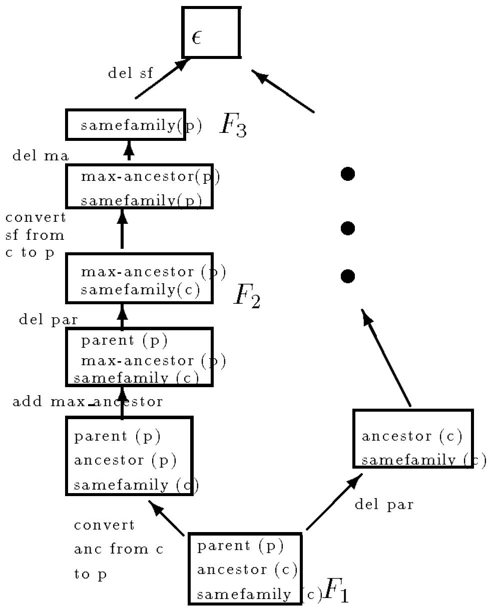

Figure 2: A portion of the space of vocabulary shifts searched by our speedup learner. Paths that generate solvers (F_2, C_o) and (F_3, C_o) are shown. The symbols p and c are used to notate primitive and computed relations. At the top is the relation with an empty extension. The space has a countably infinite branching factor, and a large, though finite, depth.

Definition 6 *The vocabulary $Vocab(T) \in VS(Vocab(T_o))$ (resp. the granularity coarsening shift $f : Vocab(T) \rightarrow Vocab(T_o)$) is a local optimum for the goal distribution D and the value function*

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

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

v if there is no $Vocab(T')$ constructible from $Vocab(T)$ by the four parametric operators such that $v(T') \succeq_D v(T)$.

We can now define the hill-climbing procedure in Table 1 for finding local optima in the vocabulary bias space. What this procedure leaves open is the instantiation of the parametric vocabulary shift operator: for instance, which specific relation should be added or deleted. To characterize the efficacy of the procedure we need to specify how this step is accomplished. The next subsection details a meta-theoretic approach to the selection of instantiated operators in the vocabulary bias space.

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. Much of the knowledge about choice of vocabulary is left implicit, which is why speedup learners cannot change their vocabulary bias in a justified way. Knowledge about the *role* of each vocabulary item in T in the goal computation 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 meta-theoretical explanations for the role of each vocabulary item in the computation of the goal distribution. One class of such explanations are *irrelevance explanations*[14]. 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 it is the case that the distinction specified by that claim is irrelevant.

In the kinship problem, observe that there are two roles played by the `parent` relation in the `samefamily` computation in F_1. Each `parent(X,Y)` fact (1) establishes first level ancestry `ancestor(X,Y)` through rule 1, and (2) establishes the set { `ancestor(X,Z)` | `ancestor(Y,Z)` derivable in (F_1, C_o)} through rule 2. We can represent this fact as follows, where $T \vdash_C f$ denotes that f is derivable from a Horn theory T with interpretation policy C.

$$\begin{aligned} \mathit{derivable\_from}(\texttt{parent(X,Y)}, F_1) &= \{\texttt{ancestor(A,B)} \mid (X = A, Y = B, \texttt{parent(X,Y)} \in F_1) \\ &\qquad \vee (X = A, F_1 \vdash_{C_o} \texttt{ancestor(Y,B)}\} \end{aligned}$$

[14]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 [Ell93] for the use of such explanations.

We can eliminate an element of the `parent` relation, if we have derived everything that can be derived from it that is essential to the `samefamily` computation. We now formally 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 7 *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 a $T' \subseteq T$ 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 expresses a property of T – that facts of the form f are unneeded for computing a proof of g in T with the interpretation policy C. Note that it is defined in terms of the provability operator $\vdash_C$. In this paper, we confine our analyses to $\vdash_C$ being SLD-resolution for definite Horn theories (resp. SLDNF-resolution for safe, stratified Horn theories with negated literals in rule bodies) 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' described in Definition 7. The definition of irrelevance is more than a notational convenience, it is a genuine meta-theoretic notion. As we shall see in the next section, it formalizes reachability properties in a dependency network of Horn clauses.

It is useful to compare our notion of irrelevance with the Anderson and Belnap notion of relevance. The statement that g is *relevant* to g in T conveys the information that knowing f in T restricts the space of possible values for g. The statement that f is *irrelevant* to g in T indicates that the value of g is insensitive to the value of f, in that even if f were eliminated from T, g would not be. 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.

$$\begin{aligned}\textbf{IC1}: &\forall \mathtt{X}, \mathtt{Y}, \mathtt{M}, \mathtt{N}\ derivable\_from(\mathtt{parent(X,Y)}, F_1) \subseteq F_1 \\ &\Rightarrow irrelevant(\mathtt{parent(X,Y)}, \mathtt{samefamily(M,N)}, (F_1, C))\end{aligned}$$

This is a conditional irrelevance claim; it states that if the condition on the right hand side is met, then it is possible to derive all `samefamily` goals in a new theory in which no `parent` fact can be derived. To apply Definition 7 to check the irrelevance claim, we augment F_1 to make the condition on the right hand side of the claim true. If the condition is of the form $x \in T$ or $x \subset T$, we add x to T to achieve it. This is a minimal revision (in the sense of subset inclusion) of T to make the condition true. Then the subset construction is applied to generate the reduced T'.

We now relate computational properties of $T = reduce(F_1, \textbf{IC1})$ to that of F_1. Intuitively, the reduction of F_1 by **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 to reduce, since all goals are solved by looking up `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. This is signaled by the appearance of an existential variable Z that occurs only in the antecedents of a rule defining the goal predicate in F_1. The corresponding irrelevance claim is:

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

Reduction of T by **IC2** yields the desired restriction on the `ancestor` relation which serves as the definition of the `max_ancestor` relation. To see this, note that we can conclude the following from **IC2** and the definition of *reduce*.

$$\begin{aligned}\texttt{ancestor}(\texttt{Y},\texttt{Z}) \in reduce(T, \mathbf{IC2}) \equiv \\ \texttt{ancestor}(\texttt{Y},\texttt{Z}) \in T\ \ \wedge\ \nexists \texttt{X}\ (\texttt{ancestor}(\texttt{X},\texttt{Y}) \in T \wedge \texttt{ancestor}(\texttt{X},\texttt{Z}) \in T)\end{aligned}$$

What this states is that if an `ancestor(Y,Z)` fact survives the reduction process, then the LHS of the **IC2** must not hold. That is, there must not be an X such that $\texttt{ancestor}(\texttt{X},\texttt{Y}) \wedge \texttt{ancestor}(\texttt{X},\texttt{Z})$. Using transitivity of `ancestor`, we can eliminate the last conjunct (`ancestor(X,Z)`) in the above definition to yield the following intensional description of the remaining `ancestor` facts. This is the required new definition.

$$\texttt{ancestor}(\texttt{Y},\texttt{Z}) \in reduce(T, \mathbf{IC2}) \equiv \texttt{ancestor}(\texttt{Y},\texttt{Z}) \in T \wedge\ \nexists \texttt{X}\texttt{ancestor}(\texttt{X},\texttt{Y}) \in T$$

It should be noted that **IC2** with `ancestor(X,Y)` substituted for `ancestor(Y,Z)` on the RHS of the implication is not a sound irrelevance claim. To prove this, we make use of the fact that `ancestor` is not symmetric. Note that the structure of the irrelevance explanations yield 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 therefore adopt the following scheme.

1. Generate those irrelevance claims for the goal class that improve computational properties of the formulation. 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 generated irrelevance claims. This step implements the shift suggested by the irrelevance claim. The reduction construction produces simplified theories that satisfy the reconstruction 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

Before we describe methods for generating irrelevance claims as well as reducing theories using them, we present a technique for verifying irrelevance claims. Two generation schemes are presented – one instantiates irrelevance schema that capture standard patterns of irrelevance in 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. Two reduction schemes are presented for Horn theories – one reduces a theory with respect

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 = $\{p \mid p$ is a path in S 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 2: 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.

to unconditional irrelevance claims, the other is an incremental procedure for reducing conditional irrelevance claims. We finally put all components together to generate a system that automatically shifts vocabulary bias. We present experimental results obtained by running the system on reachability computations in Gridworld [TR89, Coh90].

4.1 Verifying irrelevance claims

From Definition 7, 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 2 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 C has negation-by-failure and $T \vdash_C g$.
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 [Sub89] more lemmas from Definition 7, 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 2 to evaluate the truth of claims involving atomic facts and goals.

Algorithm *verify* in Table 2 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 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.

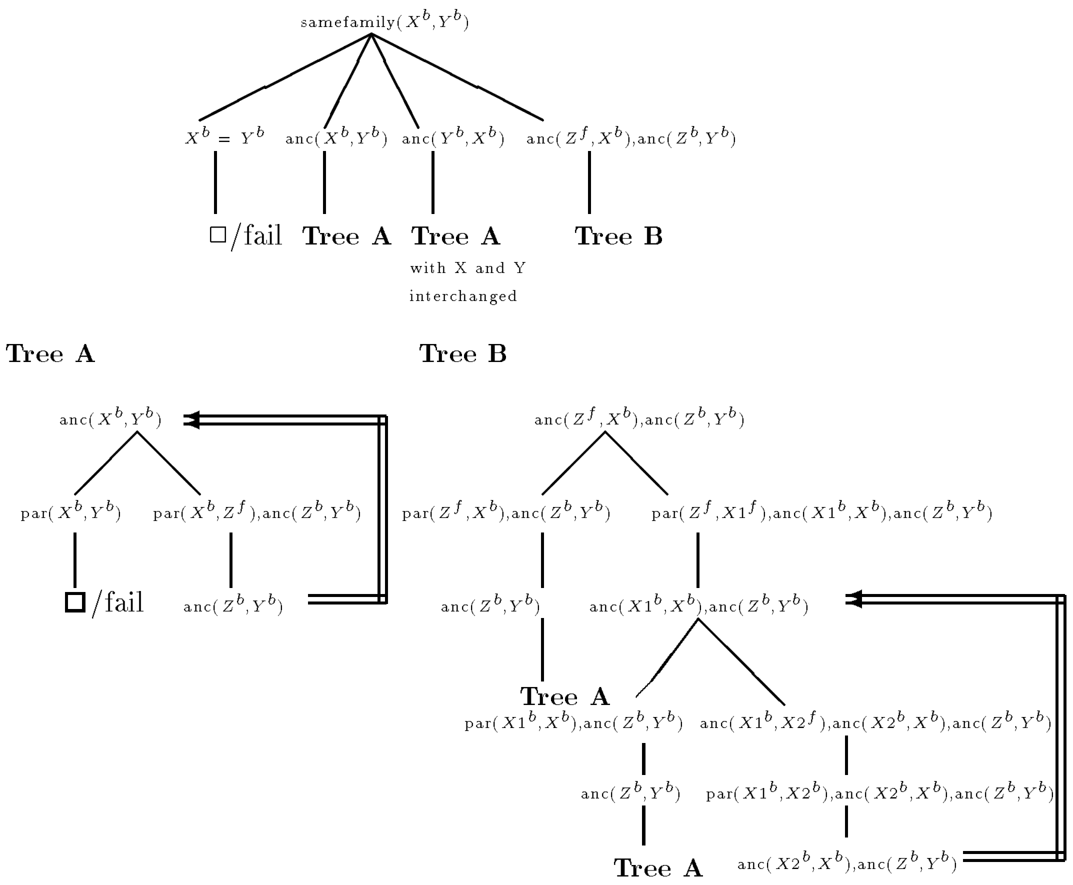

Figure 3: The C-ordered search-graph generated by a depth-first backward chainer with the standard interpretation policy C_o [rules from top to bottom and antecedents from left to right] for the `samefamily(`X^b`,`Y^b`)` goal in the formulation F_1. `anc` and `par` are abbreviations for `ancestor` and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

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[15] the theory $T = \{g(X) : -f(X), g(X) : -h(X), f(a), h(b)\}$. 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) 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 5 *Algorithm $verify(f, g, (T, C))$ is sound for definite Horn theories T, for atomic g's and f's with all variables in f annotated as free.*

Proposition 6 *Algorithm $verify(f, g, (T, C))$ is complete for definite Horn theories T for which all solutions to g can be generated finitely.*

verify(f,g,(T,C)) is also 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 7 *Algorithm $verify(f, g, (T, C))$ 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.*

4.2 Generating irrelevance claims

The verification technique above provides unification and arithmetic constraints under which claims of the form *irrelevant*$(f, g, (T, C))$ can hold. We now turn to the generation of conditional irrelevance claims of the form $R \Rightarrow$ *irrelevant*$(f, g, (T, C))$ where R is a formula involving predicates in T. We describe two general schemes for generating conditional irrelevance claims. Such claims instantiate `add` and `del` operators in the vocabulary bias space. The first scheme involves instantiating lemmas satisfying Definition 7 that identify standard patterns of irrelevance in Horn theories. The second scheme identifies transformations in the C-ordered search graph of g to reduce computation.

[15] I am indebted to an anonymous reviewer for this example.

4.2.1 Theoretical methods: irrelevance schemas

If we derive everything necessary for the goal from a predicate class `p`, then we can throw `p` away. Therefore, one general scheme for deriving irrelevance claims in a theory T for a goal class *Goal* can be stated as follows.

$$\mathbf{S1} : derivable\_from(P,(T,C)) \subseteq T \wedge P \not\equiv G \Rightarrow irrelevant(P,G,(T,C))$$

where

$$derivable(P,(T,C)) = \{Q\theta \mid (Q : -C_1,\ldots,C_m) \in T, \theta = mgu(P,C_i) \text{ for some } i, T \vdash_C Q\theta\}$$

$mgu(X,Y)$ stands for the most general unifier of the terms X and Y. $X\theta$ represents an instance of the term X with the substitution θ. *derivable_from* consists of the set of facts in T that directly require P for their derivation. This set is not deductively closed. The LHS states that facts directly derivable from P are already present in T and that P itself is not the goal relation. If we have explicitly derived everything that P can directly derive, then we no longer need P in T. We can prove this schema using Definition 7.

Proposition 8 S1 *satisfies Definition 7.*

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)`[16]. The calculation of the set `derivable_from(parent(X,Y),`(F_1, C_o)`)` can be performed using the general definition of `derivable_from` shown above. The irrelevance claim **IC2** can also be derived as an instance of **S1** using rule 2 of F_1.

If we can generate a complete set of schemas like **S1** to identify all irrelevancies in a formulation, then 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. [O.S87] showed that this is undecidable. So we can never write down a complete set of irrelevance schemas. We now turn to methods for augmenting the set of irrelevance claims generated by schema instantiation.

4.2.2 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. Most speedup learning algorithms can be treated as specific schemes for such transformations of a computation graph.

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 [Kor87], macro rules [TMKC86] and chunks [NR87]. To reduce the effective branching factor we can (1) prune unsuccessful branches in S and (2) order branches in S so that they are explored in decreasing ratio of $\frac{p}{c}$ where p is the empirically estimated probability that the branch will succeed for a goal instance, and c is the cost of generating the branch [KS75]. The ordering of branches by

[16] Technically, we treat `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.

the Simon and Kadane criterion can be implemented by changing the interpretation policy C or be statically compiled by ordering the rules and antecedents in rules in T.

We reduce the size of the C-ordered search graph S by identifying and eliminating irrelevant computational steps.

Definition 8 *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 not used anywhere. Substitutions generated by the solution of l do not contribute to a solution of the root node.*
2. *its result is already available. Substitutions for l are generated at node n_s are contained in those generated in n_f that occurs earlier in a depth-first traversal of S.*

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 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 step s.
      If l shares the thread variables in V with literals $l_1, \ldots l_m$ in n_s
         generate the new relation $t(\vec{X}) = l_1 \wedge \ldots \wedge l_m$
         where $\vec{X} = [\cup_{i=1}^{m} vars(l_i)] - \{V\}$.
      The irrelevance claims are $t(vecX) \in T \wedge C(s) \Rightarrow irrelevant(l_i, g, (T, C))$
         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 $C(p_f) \Rightarrow irrelevant(\wedge_{l \in p_f}, g, (T, C))$
      where $C(p_f)$ consists of constraints on all the variables in the literals on the path.

Table 3: 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.

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 under goals drawn from the distribution D. We implement a very special case of this definition, called *thread variable elimination*, that is syntactically cheap to detect. We present this technique via a simple example on a non-recursive definite Horn theory. Thread variable elimination constructs shortcuts in the computation graph by eliminating intermediate nodes with large branching factors. Consider the definite Horn theory $T = \{s(W) : -p(X, Y), q(Y, Z), r(Z, W).\}$ where p, q

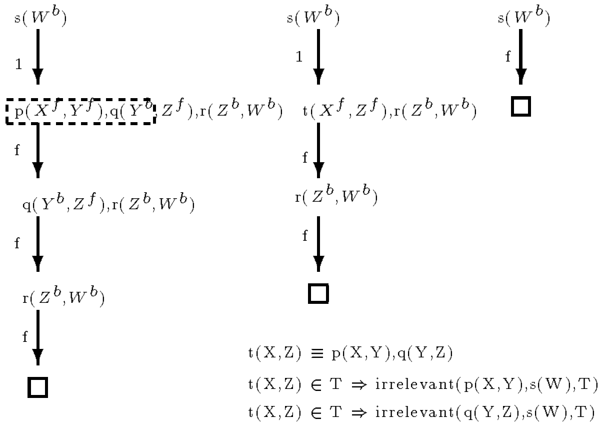

Figure 4: Transforming computation graphs by thread variable elimination. The C-ordered SLD graph S of the original theory is shown on the right. In the middle is a transformation that shortcircuits S by eliminating the thread variable Y. The irrelevance claims that implement this transformation are shown below it. The graph on the left is another transformation to S by eliminating the thread variable Z.

and r are primitive relations and s is the goal relation. The C-ordered search graph for $s(W^b)$ and two possible shortcuts of the only branch in the graph are shown in Figure 4. The branching factor at the first and second intermediate nodes from the root of the computation graph is proportional to the sizes of the primitive relations p and q.

Definition 9 *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.*

In our example, variables Y and Z are thread variables. We can eliminate the thread variable Y by introducing a new primitive relation t that precomputes the conjunction of p and q. Since there are no other goals of interest, the primitive relations p and q can are thrown away. This transformation is explained by the following irrelevance claims.

$$t(X,Z) \in T \Rightarrow irrelevant(p(X,Y), s(W), T)$$
$$t(X,Z) \in T \Rightarrow irrelevant(q(Y,Z), s(W), T)$$

The computational impact of introducing this shortcut and deleting the component steps, is the reduction of the proof height of goals of the form $s(W^b)$ by 1 and the elimination of a node in the computation graph where the effective branching factor is the set of possible instantiations for Y from p. The irrelevance claims have a very nice structure – they directly suggest what relations to add and delete. The second transformation in Figure 4 eliminates both thread variables Y and Z, and precomputes the entire s relation. Three irrelevance claims can be generated: one for each of p, q and r.

Thread variables are syntactic signals for a special case of irrelevant computation. Our method generates irrelevance claims that allow for the exploration of the tradeoff between space and time costs in a theory. It allows for the precomputation of various definable relations in the theory to speed up a goal class. The method of generating irrelevance claims by eliminating thread variables can be recursively applied to any state-space search problem formulated as a reachability computation.[17] The vocabulary shift in the kinship case presented in this paper is an instance of recursive thread variable elimination. In particular, **IC2** is a recursive variant of the non-recursive thread variable elimination rules presented here.

The algorithm in Table 3 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 at a node into a single pre-computed relation. It deletes the relation corresponding to a component conjunct as long 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. Both operations on the computation graphs eliminate irrelevant computation with respect to the goal class g, and lead to the construction of more efficient theories of coarser granularity. We can show that the irrelevance claims generated by the algorithm are correct using Definition 7. We can show that irrelevant computation is eliminated as per Definition 8 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 empirical investigations, described in Section 4.4, in the class of reachability computations suggest that the algorithm generates all the useful irrelevance claims for transitivity-style recursions.

Proposition 9 *Algorithm generate(g,(T,C)) is sound but not complete.*

4.3 Reducing theories by irrelevance claims

Once we have an irrelevance claim of the form $irrelevant(f, g, T)$, we can reduce a theory T by that claim to generate a new theory T' which satisfies Definition 7. 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.

Informally, to reduce a theory by the irrelevance claim $irrelevant(f, g, T)$ we need to produce a subset of T' that derives g without deriving f, using $\vdash_C$. There are many possible reductions of T that respect Definition 7. We need to pick some preference ordering on them so that our algorithm considers them in that order. Intuitively, we would like to make the fewest changes to T to prevent the derivation of f in it. This is because we wish to expend the least amount of effort in revising T. We translate the informally stated preference above into the following formal criterion: construct the largest (in the sense of set inclusion) subset T' consistent with Definition 7 in the reduction process.

Consider a theory $T = \{g(X) : -a(X), g(X) : -b(X)\}$ with two primitive relations a and b and a goal relation $g(X^f)$. We can use algorithm *verify* to show that $irrelevant(a(Y), g(X^f), (T, C))$. There are two possible reduction procedures for this definite Horn theory: (1) remove every clause in T in which a literals occur positively. Thus all ground facts about a are eliminated from T. (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 and keeping all the ground a facts. Thus the reduction

[17] 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.

```
reachable(Start,End) :- one_step(Start,Int),reachable(Int,End).
```

procedure *reduce(T,irrelevant(f,g,(T,C))*

Input: a Horn theory T and an 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. $T' = T$ - $\{c \mid c \in T, f$ occurs positively in $T\}$.

b. if f is primitive, then $T' = T$ - $\{c \mid c \in T, f$ occurs negatively in $T\}$.

c. if f is derived, then construct a list L of clauses with heads that unify with f.

Initialize T' to be T.

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

set T' to be *reduce*$(a_1 \wedge \ldots \wedge a_m, g, (T', C))$

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

Non-deterministically generate one of:

a. T' = *reduce(T,irrelevant*$(f_1,g,(T,C))$

b. T' = *reduce(T,irrelevant*$(f_2,g,(T,C))$

Table 4: The reduction procedure *reduce(T,irrelevant(f,g,(T,C)))*.

of T by this irrelevance claim involving a primitive relation, 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 $T = \{f(X) : -p(X), q(X).f(X) : -r(X).g(X) : -h(X)\}$ where p, q and r are primitive relations, f is a computed relation and g is the goal relation. f is the relation to be eliminated. The general principle behind reducing T with respect to the claim $irrelevant(f, g, (T, C))$ where f is a computed relation, is to cut every successful path in the computation graph that contains a literal that unifies with f. Each path can be cut at one or more points, yielding a combinatorial number of revision possibilities. (1) We can eliminate every rule in which f occurs positively. This deletes the two rules that derive f in T keeping everything else. This breaks the derivation of f at the very last step, since all the antecedents to establish f remain derivable. (2) For every rule in which f occurs as a consequent, we can recursively eliminate the antecedents. In this example, we can eliminate one of p or q (using Lemma 1. of Section 4.1) since they are primitive relations, but not both, since we need the largest subset of T that does not entail f. If p had been a computed relation, we can recursively trace its primitive support set. This is the source of the combinatorial explosion in the number of minimal revisions of T.

Proposition 10 *The reduction procedure in Table 4 satisfies its specifications. That is, upon termination it produces* $T' = reduce(T, irrelevant(f, g, (T, C))$ *such that* $T' \not\vdash_C f$ *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 11 *The problem of obtaining the least cost (with respect to a value function v) revision of* T *with an irrelevance claim is NP-hard.*

So far we have discussed reduction of theories by unconditional irrelevance claims. In the kinship example, both irrelevance claims are conditional in nature. To implement the shift suggested by a conditional irrelevance claim, we augment T by the propositions specified in the condition. As an example, consider reduction by **IC2** of the formulation F_1, after reduction by **IC1**. When we find `ancestor(X,Y)` and `ancestor(Y,Z)` in the database, we add `ancestor(X,Z)` and delete `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 Prolog code for extensional reduction is in Appendix IV.

We now discuss two important properties of the conditional irrelevance reduction procedure.

1. *Order of reduction*: When there is more than one irrelevance claim, there is a choice to be made regarding the order of reduction. One natural question to ask is whether this order makes a difference to the end result; i.e. whether reductions satisfy the Church-Rosser property. The answer is that in the cases where it is meaningful to reduce in any order, reductions do not obey the Church-Rosser property. When the claims are independent, then the order is irrelevant. In our example, if we start with a database containing `parent` facts alone, the analysis of the preconditions of **IC1** and **IC2** show that **IC1** needs to be applied before **IC2**. So there is only one order of reduction. Suppose for instance that the effect of an irrelevance claim is to take the transitive reduction of a relation and the effect of another claim is to take the transitive closure of the same relation. It is clear that the order of reduction makes a significant difference to the end result.

2. *Termination of reduction*: How can we know that the reduction process terminates? We can reason that the reduction process terminates on the kinship example by using the fact that the databases are finite and that the relations are acyclic. However, the fact that checking termination reduces to the halting problem, we have fundamental limits on inferring it in the general case.

4.4 Experimental results

We have implemented the algorithms discussed in the previous subsections and constructed a vocabulary bias shifter. The shifter has three major components: (1) Irrelevance claim generator with schema-based generation of irrelevance claims and the empirical method of generating claims by thread variable elimination, (2) Formulation reducer that revises the formulation in accordance to the given set of irrelevance claims, and (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. Its sample complexity is polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ [SH92]. The formulation evaluation function v used is a linear combination of coverage, time and space cost. By varying the weights of the different components of the evaluation function, we can have the evaluator yield different formulations as local optima in the formulation space. We have experimented with two different policies for sequencing the reduction and evaluation stages. We evaluate a formulation after it has been reduced by one or all of the generated irrelevance claims. The latter policy corresponds to taking directed macro-steps in $VS(Vocab(T_o))$. In the kinship example explained in this paper, we illustrated this latter scheme. In the Gridworld domain considered next, the former policy was employed and it worked exceedingly well.

We performed scale 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 [TR89] and studied in the context of learning control rules for speedup in [Coh90]. The world is a square n by n grid of points $p_i, 1 \leq i \leq n^2$, represented by the `conn(X,Y)` relation, which records 1-step connections in the grid. `conn` is a symmetric and transitive predicate. The `reachable(X,Y,D)` predicate is recursively computed from `conn` using a definite Horn theory (a full formulation is shown in Appendix IV) and represents the fact that there is a path of length D that connects X and Y in the grid. The goal class is `iterate_reachable(`X^b`,`Y^b`, `D^f`)` and our theory computes it by attempting to find paths of increasing length from 1 to n. The complexity of this goal class varies exponentially with the grid size n; a fact reflected in Figure 5. We use the technique of thread variable elimination to generate the one irrelevance claim for this domain. If `conn(X,Y)` and `conn(Y,Z)` are present in the theory, we eliminate `conn(X,Y)` by adding all `conn(X,Z)` facts such Y is one-step connected to Z (via the `conn` relation). This irrelevance claim applies to all grid sizes. So we discover this using both the empirical and the schema-based irrelevance claim generation method for a small grid size (n=2) and then reduce formulations for $2 \leq n \leq 8$ with it. In all cases, the reducer rewired the `conn` relation so that all points in the grid are 1-step connected to a common point in the grid. 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 policy 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 $2 \leq n \leq 8$. We then tested the following hypothesis: the reduced theory is faster than the original theory for computing reachability over 1000 point pairs (x, y) drawn from a uniform distribution on an n by n grid. No censoring of data points was used, and 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 Wilcoxson signed rank test was used for judging the significance of the speedup. For $2 \leq n \leq 8$, the new theory outperformed the old theory (significance level of test was 0.01%). More importantly, the complexity of the goal class under the new theory grows linearly with grid size. 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 wanted to check 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 ($2 \leq n \leq 8$), the formulation converged to a set of facts where each grid point was 1-step connected to a common grid point.

3. *Performance of the extensional reducer*: An important question 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

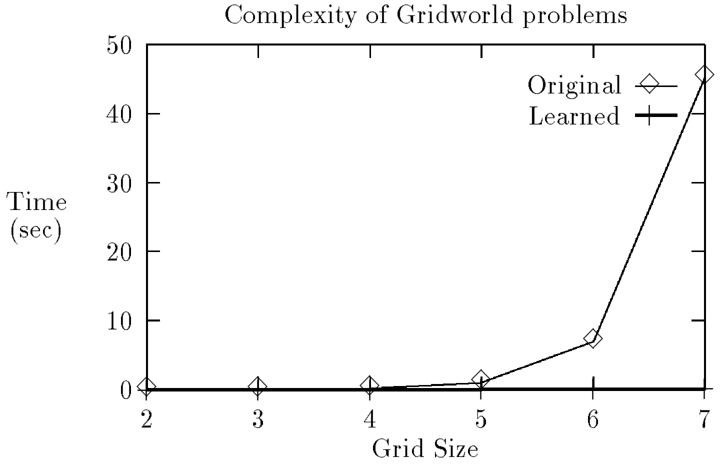

Figure 5: Performance of the original domain theory and the new theory generated by irrelevance minimization as the complexity of Gridworld is scaled up.

of implementing the "best" reduction, until there are no more reductions to process. This policy allows it to continue even if the hits a local minimum in the formulation space until all irrelevancies specified by the claims are eliminated.

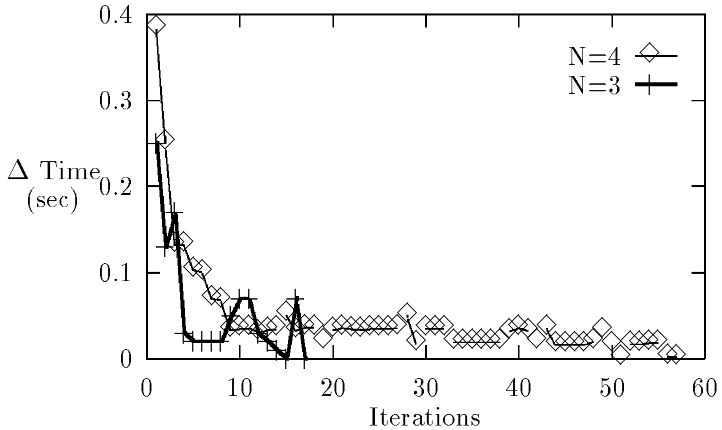

Figure 6: Performance of the gradient descent bias shift algorithm. All times are scaled by 10 for N=3.

4.5 Discussion

The kinship example in this paper is a reformulation of an equivalence relation as a partition; 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 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 [Web93] 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 8), 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.[18] — these are points in the vocabulary space $VS(Vocab(T_o))$. 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 for a long while. In 1980, Korf [Kor80] characterized the nature of the information transformations that occur during shifts of vocabulary. He defined a set of rewrite rules on theories to set up a space of possible shifts. His theory explained vocabulary shifts in the Tower of Hanoi puzzle as well as some examples from floor planning. In 1981, Jack Mostow designed a program that reformulated advice in the game of Hearts: he devised 400 transformation rules to generate the space of reformulations, which included shifts of vocabulary. These transformation rules were a set of relational rewrite rules of the form: $exp1 \rightarrow exp2$ where $exp1$ and $exp2$ were first-order formulas. These rules formed the generator for the space of new theories and computational constraints were used as testers to determine if a better theory had been found. The main disadvantage of this approach was that there were too many rewrite rules even in domains of medium complexity (Hearts) and the problem of deciding the order in which to apply them proved formidable. Another disadvantage of the approach was that there was no principled way of generating these rewrite rules from more global considerations. In 1983, Utgoff [Utg86] attacked automating shifts of vocabulary bias in the inductive context with the Lex [MTMB82] 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. The addition of new vocabulary to an inductive system to make prediction efficient was studied by Fu & Buchanan [FB85]. This work proposed two general heuristics and was tested empirically in the domain of medical diagnosis. In 1986, Richard Keller [Kel87] introduced a scheme for the

[18]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.

addition of new vocabulary items into the Lex solver to make the solution of special query classes efficient. He used empirical methods for testing the efficacy of a given representation for a class of goals.

More recently, there has been a flurry of work on the problem of automatic introduction of new vocabulary to make problem solving and machine learning more efficient. Most of them focus on very special classes of deductive and inductive abstractions. Patricia Riddle, [Rid88] working with Saul Amarel, has attempted to automate the formation of deductive abstractions of the missionaries and cannibals problem. Lowry [Low88] has developed abstraction methods based upon the theory of abstract data types to synthesize algorithms from specifications. Muggleton [Mug88] has devised a machine learning scheme that introduces new predicates to make compact generalizations of instances. Flann [Fla88] reformulates theories expressed in structural terms into functional terms to make a recognition task more efficient. Flann uses examples to guide selection of useful functional terms. The generation of functionally useful classes by induction from problem solving traces is the theme in Jeff Schlimmer's work on representation change [SP88]. Recent work in abstraction by Knoblock [KTY91] in the context of Strips planning automatically generates abstraction of operators by ordering operator preconditions. We can re-express the justifications for precondition ordering as meta-theoretic conditional irrelevance statements. framework of conditional irrelevance

5.2 Conclusions

Research in automating shifts of vocabulary bias in speedup learning is still exploratory in nature because our knowledge is too preliminary to even formulate the appropriate questions, let alone solve them. 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. The framework attempts to combine a way of looking at things that has powerful heuristic value with a collection of mathematical definitions and theorems that can be used for rigorous derivation of results.

We introduced the need for meta-theoretic explanations of the role of a vocabulary item for justified shifts of bias. We discussed a special class of such explanations, based on irrelevance. Irrelevance explanations are special because they prescribe changes to a theory and its vocabulary that impacts its computational properties. They give us the basis for eliminating certain relations, as well as adding new definable relations to the vocabulary of a theory. The crux of our paper was the transformation of irrelevance explanations into prescriptions for additions of new relations and deletion of existing relations to improve computational efficiency. This is done by meta-theoretic reduction inferences that modify a given theory so that the irrelevance claims are no longer true of the new theory. This process is guided by a local optimisation principle called the *irrelevance principle* whose informal statement is: the minimization of distinctions with respect to a goal distribution minimizes computational effort in the solution of these goals.

The chief algorithmic result is the development of a hill climbing search of a well formulated bias space aided by irrelevance claim generators and formulation reducers. The thread variable elimination technique for the generation of irrelevance claims by analyzing C-ordered search graphs of the computation of the goal class was presented. Reduction methods that delete relations and introduce new definable relations were developed in the context of definite (and stratified) Horn theories.

Our theory has been empirically verified by constructing a prototype of a vocabulary bias shifter employing the algorithms developed here. It has been tried on the kinship example, Gridworld and its variants, and other reachability computations including the generation of Thevenin equivalents of analog circuits. 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 shifts of vocabulary bias in different domains as instances of a powerful invariant in granularity shifts: the most economical description is the one that uses concepts of the largest granularity consistent with the correctness constraints. 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.

References

[Bun90] W. Buntine. Myths and legends in learning classification rules. In *Proceedings of AAAI-90*. Morgan Kaufmann, 1990.

[Coh90] W. Cohen. Learning approximate control rules of high utility. In *Proceedings of ML90*, pages 268–276, 1990.

[Ell93] T. Ellman. Abstraction by approximate symmetry. In *Proceedings of IJCAI-93*, Chambery, Italy, August 1993.

[FB85] L-M. Fu and Bruce G. Buchanan. Learning intermediate knowledge in constructing a hierarchical knowledge base. In *Proc. of IJCAI-85*, pages 659–666, Los Angeles, CA, August 1985. IJCAI, Inc.

[Fla88] N. Flann. Improving problem solving performance by example-guided reformulation of knowledge. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[Gor90] D. Gordon. *Active bias adjustment for incremental supervised concept learning.* PhD thesis, University of Maryland, 1990.

[GS92] D. Gordon and W. Spears. Is consistency harmful? In *Proc. of the Workshop on biases in inductive learning, ML-92*, 1992.

[Hau90] D. Haussler. Quantifying inductive bias: Ai learning algorithms and valiant's learning framework. In J. Shavlik and T. Dietterich, editors, *Readings in machine learning*, pages 205–227. Morgan Kaufmann, 1990.

[Hob85] J. Hobbs. Granularity. In *Proceedings of IJCAI-85*. Morgan Kaufmann, 1985.

[Kel87] R.M. Keller. *Learning in Context.* PhD thesis, Rutgers University, November 1987.

[Kor80] R. E. Korf. Toward a model of representation changes. *Artificial Intelligence*, 14(1):41–78, August 1980.

[Kor87] R. Korf. Planning as search: a quantitative approach. *Artificial Intelligence*, 33(1):65–88, 1987.

[KS75] J.B. Kadane and H.A. Simon. Optimal problem-solving search: all-or-none solutions. *Artificial Intelligence*, 6, 1975.

[KTY91] C. A. Knoblock, J.D. Tenenberg, and Q. Yang. Characterizing abstraction hierarchies for planning. In *Proceedings of AAAI-91*, 1991.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming (second extended edition)*. Springer-Verlag, 1987.

[Low88] M. Lowry. Problem reformulation and adt. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[Mit90] T. Mitchell. The need for biases in learning generalizations. In J. Shavlik and T. Dietterich, editors, *Readings in machine learning*, pages 184–191. Morgan Kaufmann, 1990.

[MTMB82] Utgoff P. E. Mitchell Tom M. and Banerji R. B. Learning by experimentation: Acquiring and refining problem- solving heuristics. In *Machine Learning*. Tioga, 1982.

[Mug88] S. Muggleton. Constructive induction in first-order logic. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[NR87] J.E. Laird A. Newell and P.S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[O.S87] O.Shmueli. Decidability and expressiveness aspects of logic queries. In *Proc. 6th ACM PODS*, pages 237–249, 1987.

[Pro92] Foster Provost. *Policies for the selection of inductive bias in machine learning*. PhD thesis, University of Pittsburgh, 1992.

[RG87] S. Russell and B. Grosof. A declarative approach to bias in concept learning. In *Proceedings of AAAI-87*, pages 505–510. Morgan Kaufmann, 1987.

[Rid88] P. Riddle. 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, June 1988. Philips Laboratories.

[Ros83] P. Rosenbloom. *The chunking of goal hierarchies: a model of practice and stimulus-response capability*. PhD thesis, Carnegie-Mellon University, 1983.

[SG87] D. Subramanian and M.R. Genesereth. The relevance of irrelevance. In *Proceedings of IJCAI-87*, Milan, Italy, August 1987.

[SH92] D. Subramanian and S. Hunter. Measuring utility and the design of provably good ebl algorithms. In *Proceedings of ML-92*, 1992.

[SP88] J. Schlimmer and A. Proca. Learning, problem-solving and representation change. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, New York, June 1988. Philips Laboratories.

[Sub89] D. Subramanian. *A Theory of Justified Reformulations*. PhD thesis, Stanford University, March 1989.

[TMKC86] R. Keller T. Mitchell and S. Kedar-Cabelli. Explanation-based learning: A unified view. *Machine Learning*, 1(1):47–80, 1986.

[TR89] M. Tambe and P. Rosenbloom. Eliminating expensive chunks by restricting expressiveness. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 731–737. Morgan Kaufmann, 1989.

[Ull89] J.D. Ullman. *Principles of database and knowledge-based systems, Vol II.* Computer Science Press, 1989.

[Utg86] P.E. Utgoff. *Shift of Bias for Inductive Concept Learning.* PhD thesis, Rutgers University, 1986.

[Web93] Adam Webber. *Optimizing functional programs by reasoning about irrelevant computation.* PhD thesis, Cornell University, 1993.

Appendix I: Derivation of cost for F_1 on D

We now analyze the cost of average cost of solving goals in D using F_1. Let $C(sf_{bb}, F_1)$ denote the average cost of solving a goal in the distribution. Let i be the cost of reducing a goal to its subgoals by a single inference step. Let d be the cost of looking up a fact and s be the cost of storing a fact. These costs are expressed in common units (say, dollars) so that we can add them directly. The cost of a formulation is the average cost of solving goals from the distribution as well as the space cost associated with storing the ground facts in the formulation. We denote the size of a ground relation `r` as *size(*`r`*)*.

$$\begin{aligned} Cost(F_1, D) &= Space\_cost(F_1, D) + Time\_cost(F_1, D) \\ &= size(\texttt{parent}).s + C(sf_{bb}, F_1) \end{aligned}$$

We now describe the derivation of the cost recurrences for $C(sf_{bb}, F_1)$ by our cost analyzer. We expect the average cost to be a function of the distribution of the bindings for the `samefamily` goal in D, as well as the topology of the `parent` relation. Let p_{xx} stand for the probability that a goal drawn from D has the form `samefamily(X,X)`. Let p_{xy} (resp. p_{yx}) stand for the probability that a goal from D is solved by finding an ancestry chain from `X` to `Y` (resp. from `Y` to `X`). $C(A_{bb})$ stands for the average cost of solving an `ancestor` goal with both its arguments bound, while $C(A_{fb})$ stands for the cost of solving it with its first argument free and its second argument bound. Our solver considers rules in sequence and solves conjunctions from left to right. c is the average length of a chain in the `parent` relation. The cost rule below is mechanically derived by our cost analysis system.

$$\begin{aligned} C(sf_{bb}) \;\; = \;\; & p_{xx}[i] + p_{xy}[2i + C(A_{bb})] + p_{yx}[3i + 2C(A_{bb})] + \\ & (1 - p_{xx} - p_{xy} - p_{yx})[4i + 2C(A_{bb}) + C(A_{fb}) + cC(A_{bb})] \end{aligned}$$

Now the analyzer attempts to derive recurrences for $C(A_{bb})$ and $C(A_{fb})$ using the rules defining the `ancestor` predicate. It generates the following set of equations which it recognizes are not solvable,since the term $C(A_{bb})$ occurs in both the RHS and the LHS of the equation indicating then need for developing a recurrence. k is the average number of solutions to an `father`(X^b, Y^f) goal. For each primitive relation, the system computes average number of solutions for each binding category (for binary relations, there are four categories bb, bf, fb and ff).

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

In a non-structural theory, where the basis of the recursion is not the length of the input expression, the system has to rely on the user to restructure the equations by explicitly identifying the recurrence basis. So 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 with N intermediate ancestors. The base case of the computation occurs with the reduction to `parent(`X^b`,`Y^b`)` by the first rule and a check to see if the parent fact occurs in the database – this costs $i + d$ units. The recursive case with N intermediate ancestors consists of reduction of the goal by the second rule into two conjunctive ancestor subgoals – F_{bf} and A_{bb}. If no solutions to the first conjunct are found, the computation terminates and a failure to solve the top level goal is reported. The cost of solving the conjunction produced by the second rule is the cost of solving the first conjunct plus the cost of solving the second conjunct with bindings obtained from the first. This process can repeat an average of k times with *different* intermediate ancestors chosen during the solution of the first conjunct. The average case cost equations are shown below. Note their relationship to the mechanically generated equation above.

$$\begin{aligned} C(A_{bb}, 0) &= i + d \\ C(A_{bb}, N) &= (i + d) + i + k[d + C(A_{bb}, N - 1)], \; N > 1 \end{aligned}$$

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 `ancestor(`X^f`,`Y^b`)`, unlike the case where both arguments are bound, there are multiple solutions for X. In [SH92], we describe how 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 the first `ancestor` rule. For the subsequent solutions, the cost $C(A_{fb})$ per solution derived by the analyzer using the transitivity rule on `ancestor` 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:

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

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. The average cost of a goal in D with the formulation F_1 is $O(ck^N)$. Thus it is linear in the length of the `parent` chains and exponential in the local branching factor of `parent`.

Appendix II: Constructing C-ordered search graphs

We now proceed to formally describe C-ordered search graphs. Since we work with definite clause Horn theories T, we use SLD-resolution to prove goals drawn from the problem distribution. The goals considered are also definite; they are a conjunction of atomic literals, from the domain of T.

Definition 10 (Lloyd) *We will say that G' is derived from G and the definite clause $H : -A_1, \ldots A_n$ by SLD-resolution if G is the goal $G_1, \ldots, G_m, \ldots G_k$, σ is the most general unifier of H and G_m, and G' is $(G_1, \ldots G_{m-1}, A_1, \ldots, A_n, G_{m+1}, G_k) \cdot \sigma$, 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.

1. `ancestor(X,Y) :- father(X,Y).`

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

Consider the ground goal `ancestor(a,b)`. We can use SLD-resolution with rule 1 to obtain the new goal `father(a,b)` with most general unifier $\{X/\mathtt{a}, Y/\mathtt{b}\}$. SLD-resolution with rule 2 yields the new conjunction `ancestor(a,Z),ancestor(Z,b)` with most general unifier $\{X/\mathtt{a}, Y/\mathtt{b}\}$. We will 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 11 *Let T be a set of definite clauses and G a definite goal. A C-ordered search graph for $T \cup \{G\}$ is the directed graph defined below. Each node of the graph contains a possibly empty list of definite goals.*

1. *The root node is the list containing G.*

2. *Let $A_1, \ldots, A_k$ ($k \geq 1$) be a node in the tree. If there exists another node in the tree with the literal list $B_1, \ldots, B_k$ ($k \geq 1$) where the A list is identical to the B list upto variable renaming, construct a directed unification edge (double lined edges in our figures) from the node at lower in the tree to the one at the higher level in the tree.*

3. *Let $A_1, \ldots, \ldots, A_k$ ($k \geq 1$) be a node in the tree. For the atom A_1, let $R = \{A\text{:-}B_1, \ldots, B_q \in T \mid A_1$ and A are unifiable with mgu $\sigma\}$. The node has $|R|$ children: each child is a node of the form $(B_1, \ldots, B_q, A_2, \ldots, A_k) \cdot \sigma$. These are children that are produced by SLD-resolution on the first literal at the parent node using the elements of R. The edge from the node to each child is labeled by the name of the rule in R and the substitution σ used in the reduction. if A_1 is an arithmetic computation or a built-in predicate (e.g., equality) that is solved by procedural attachment by our solver, we generate the child node $\{A_2, \ldots, A_k\} \cdot \sigma$ where σ consists of the substitutions obtained by solving A_1. The edge connecting the parent and child node in this case is labeled u.*

Nodes which are the empty clause have no children. Nodes for which no rule head in T unifies with the selected literal also have no children. Because of the cycle detection step 2 above, 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 three types of branches in this tree: successful branches corresponding to successful derivations of G (these end in the empty clause), infinite branches corresponding to infinite derivations, and failed branches with non-empty leaves [Llo87]. The C-ordered search graph is the set of possible schematic proofs of G in T.

Definition 12 *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 assume that all the goals we are concerned with, in the kinship theory, have the form `ancestor(X,Y)` where both `X` and `Y` are bound. The actual constants will be provided at run-time. We notate the goal scheme using the convention followed in the database optimization literature [Ull89], by superscripting variables with b or f standing for whether they are bound or free. The definitions below show how annotations on goals are propagated by an SLD-resolution step.

Definition 13 *An annotated literal $G(X^{a_1})$ unifies with the annotated literal $G(P^{a_2})$ where X and P are annotated variables, when $G(X)$ and $G(P)$ unify by the classical rules, and 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 14 *We will say that the annotated set of goal literals G' is derived from the annotated goal literal G and the definite clause $H : -A_1, \ldots A_l$ 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 13, and G' is $(B_1, \ldots, B_l) \cdot \sigma$, 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 rules*[19]*:*

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 $\mathtt{A}x^b y^b$ with clause 1 yields the goal class $\mathtt{F}x^b y^b$. SLD-resolution of $\mathtt{A}x^b y^b$ with clause 2 produces the goal class $\mathtt{A}x^b z^f$, $\mathtt{A}z^b y^b$. This is because $\mathtt{A}x^b y^b$ unifies with $\mathtt{A}x^f y^f$: the head of clause 2 augmented with annotations, to produce $\sigma = \{X/X, Y/Y\}$ where X and Y are both annotated by b. The literal `ancestor(X,Z)` is annotated as $\mathtt{A}x^b z^f$ since X inherits the annotation from the head of the clause and Z obtains the annotation f by the definition above. The literal `ancestor(Z,Y)` obtains the annotation $\mathtt{A}z^b y^b$, since Z is bound by the previous literal and Y is bound in the head. Using Definition 14, we can generate schematic C-ordered search graphs for a goal class instead of a specific goal. A statically generated C-ordered search graphs is shown in Figure 3. Nodes that are alphabetic variants of each other are connected by double edges.

Proposition 12 *The construction of a C-ordered search graph for definite goal 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.*

Thus for predicates of bounded arities[20], the construction process is polynomial in the size of T.

Proofs of Propositions

5.3 Propositions in Section 3

Proposition 13 $\succeq$ *is a pre-order on a set V of vocabularies.*

Proof: We will demonstrate that $\succeq$ is reflexive, and transitive.

[19] Note that H acquires new annotations after being unified with G.

[20] all useful examples can be coded with predicates of small arity.

1. Reflexive: Let $v \in V$. $v \succeq v$ follows from the fact that there exists an f – the identity map on object and relation symbols on v to itself that satisfies Definition 1.
2. Transitive: Given that $v_1 \succeq v_2$ and $v_2 \succeq v_3$ for $v_1, v_2, v_3 \in V$, we can construct $f : v_1 \rightarrow v_2$ and $g : v_2 \rightarrow v_3$ by Definition 2. Since the object and the relation maps of f and g can be composed by the usual definition of function composition, we can construct $h : v_1 \rightarrow v_3$ that satisfies Definition 1. $\square$.

We provide a short introduction to definite Horn theories.

A definite clause is a clause of the form $A : -B_1, ldots, B_m$, with exactly one positive literal A at the head and the conjunction of positive literals B_i that form the body of the clause. All variables in a clause are assumed to universally quantified. A definite Horn theory consists of a finite set of definite clauses. Definite Horn theories are special because they do not contain negative literals in the clause bodies. We therefore do we not have to deal with negated subgoals. Definite theories have a unique least Herbrand model. SLD-resolution is a sound and complete inference procedure for definite theories.

The declarative semantics of a definite Horn theory is given by the usual model-theoretic semantics of formulas in first-order logic. We are interested in a special class of models, called Herbrand models. A ground term is a term not containing variables and a ground atom is a positive literal that does not contain variables.

Definition 15 *The* Herbrand universe *U_T of a definite theory T consists of the set of all ground terms that can be formed from constants and function symbols appearing in T. The* Herbrand base *of a definite theory T is the set of all ground positive literals that can be formed by using predicate symbols from T with ground terms from the Herbrand universe U_T as arguments.*

The Herbrand pre-interpretation of a definite theory T consists of the domain of discourse U_T which is the Herbrand universe of T with the constants in T being assigned themselves in U_T and if f is an n-ary function symbol in T, then the mapping f' from $(U_T)^n$ into U_T defined by $(t_1, ldots, t_n) \rightarrow f(t_1, \ldots t_n)$ is assigned to f.

Definition 16 *A Herbrand interpretation for T consists of the Herbrand pre-interpretation for T together with the assignment of $p' : {U_T}^n \rightarrow \{true, false\}$ for each n-ary predicate symbol p in T.*

Let I be a Herbrand interpretation of T and let V be a variable assignment which is a map from variables to elements of U_T. The term assignment for T with respect to I and V is as follows: each variable is given its assignment according to V, and each term is mapped to itself in U_T. Since, for Herbrand interpretations, the assignment to constants and function symbols is fixed, it is possible to identify an Herbrand interpretation with a subset of the Herbrand base. For any Herbrand interpretation, the corresponding subset of the Herbrand base is the set of all ground atoms which are true with respect to interpretation.

Definition 17 *A formula in a Horn theory T can be given a truth value* true *or* false *with respect to a Herbrand interpretation I and variable assignment V as follows.*

- *The truth value of an atomic formula $p(t_1, \ldots, t_n)$ is obtained by calculating the value of $p'(t_1, \ldots, t_n)$. That is, we check if $p'(t_1, \ldots, t_n)$ is an element of the subset of the Herbrand base of T made true by I.*

- *If the formula has the form $\neg F$, then it is true if and only if F has truth value false. If the formula has the form $F \vee G$ then it is true if and only if F or G has truth value true with respect to I and V. If the formula has the form $\forall x.F$, then it is true if for all $d \in U_T$, we have that F has truth value true with respect to I and with every occurrence of x in F substituted by d; otherwise its truth value is false.*

Definition 18 *Let I be a Herbrand interpretation of a definite Horn theory T. Then I is a model of T if every clause C in T is true in I, with respect to any variable assignment V.*

Proposition 14 *If the definite theory T has a model (or equivalently, is consistent), then it has a Herbrand model.*

Proposition 15 *A consistent definite Horn theory T has a unique least Herbrand model which is the intersection of all the Herbrand models of T.*

If a formula F is true in the least Herbrand model of T with respect to any variable assignment V, then we say that $T \models F$. A goal formula is definite if it is a positive literal or a conjunction of positive literals.

Proposition 16 *For a definite Horn theory T and a definite goal G, if $T \vdash G$ by SLD-resolution, then $T \models G$. If $T \models G$, then there exists a proof of G in T using SLD-resolution.*

This proposition states the soundness and completeness of SLD-resolution for definite Horn theories and definite goals.

Proposition 17 *For every model M_1 of T_1, we can construct a model M_2 of T_2 from $f : Vocab(T_2) \rightarrow Vocab(T_1)$.*

This construction process is outlined in the text of the paper.

Definition 19 Definability*: A relation R is definable in a definite Horn theory T in terms of the relations $S_1, \ldots S_k$ if there is a first order formula ϕ involving a new relation symbol r and the relation symbols $s_1, \ldots s_k$ in T such that if ϕ is true in the least Herbrand model of T, and the interpretations of the s_i's are the S_i's, $1 \leq i \leq k$, then the interpretation of ϕ in the model is exactly R. ϕ is the defining formula for R in T.*

The usual notion of definability is with reference to a model. Since definite theories have associated with them a unique least Herbrand model, we can substitute "model" for "definite Horn theory" in the above definition. As a simple example consider the definite Horn theory about kinship introduced in Section 2. The binary relation *grandparent* consisting of pairs (x, y) where $x, y \in \mathcal{P}$ with x being the grandparent of y is definable in terms of the *parent* relation. The defining formula is

$$\mathtt{grandparent(X,Y) :- parent(X,Z), parent(Z,Y)}$$

The relation `max_ancestor` is definable in the theory F_1 because it has a defining formula $\mathtt{ancestor(X,Y)} \wedge \neg\mathtt{parent(Z,X)}$, involving relation symbols `parent` and `ancestor` in F_1 satisfying the above definition. Note that the defining rule is not definite, it is stratified and because the negated subgoal is a primitive relation, F_1 with this defining rule has a unique least Herbrand model computed by SLDNF-resolution. The least Herbrand model of F_1 maps the parent relation symbol to the primitive parent relation, the ancestor symbol to the transitive closure of the parent relation and the max_ancestor symbol to the set of pairs (x, y) where x and y are drawn from $\mathcal{O}$ and x is the maximal ancestor of y.

Proposition 18 *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 f to the space of formulas.*

To show this formally, we extend the map $f : Vocab(T_2) \rightarrow Vocab(T_1)$ on vocabulary items to a function $f_L : L(Vocab(T_2)) \rightarrow L(Vocab(T_1))$ on formulas. The base case is as follows. Given a formula $\phi_1 = r(a_1, \ldots, a_n)$ in $L(Vocab(T_2))$, we have $f_L(\phi_1) = f(r)(f(a_1), \ldots, f(a_n)))$. The inductive cases are: for $\phi_1, \phi_2 \in L(Vocab(T_2))$, $f_L(\phi_1 \vee \phi_2) = f_L(\phi_1) \vee f_L(\phi_2)$, $f_L(\neg\phi) = \neg f_L(\phi)$, $f_L(\exists x.\phi) = \exists x.f_L(\phi)$ where x is free in ϕ.

5.4 Propositions in Section 4

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.

   Proof: $\Leftarrow$: From the RHS we can infer that there is either a subset T_1 of T such that $T_1 \vdash_C g$ and $T_1 \not\vdash_C f$, or a subset T_2 of T such that $T_2 \vdash_C g$ and $T_2 \not\vdash_C h$. To prove the LHS, we need to establish the existence of a subset S of T such that $S \vdash_C g$ and $S \not\vdash_C f \wedge h$. If $S \not\vdash_C f \wedge h$, then either $S \not\vdash_C f$ or $S \not\vdash_C h$ provided $f \wedge h$ is consistent (there is a non-empty model of T in which f and h are both true). We can exhibit $S = T_1$ or $S = T_2$ depending on whether f or h was irrelevant to g.

   $\Rightarrow$: If the LHS is true, there is a subset S of T such that $S \vdash_C g$ and $S \not\vdash_C f \wedge h$. If $f \wedge h$ is consistent with respect to T , then we can show that either $S \not\vdash_C f$ or $S \not\vdash_C h$. Therefore, we can either show $irrelevant(f, g, (T, C))$ or $irrelevant(h, g, (T, C))$ with the witness set S. $\square$.

2. $\neg irrelevant(\neg f, g, (T, C)) \Rightarrow irrelevant(f, g, (T, C))$ if C has negation-by-failure and $T \vdash_C g$[21].

   Proof: $\Rightarrow$: The LHS assures us that there is no subset S of T such that $S \vdash_C g$ and $S \not\vdash_C \neg f$. Since in C, failure to prove a literal is taken as proof for its negation, we have $S \vdash_C f$. Since $T \vdash_C g$, there is a subset S' such that $S' \vdash_C g$. This subset must be such that $S' \not\vdash_C f$ because of the LHS condition. This means that $irrelevant(f, g, (T, C))$.

3. $irrelevant(f, g_1 \wedge g_2), (T, C)) \Rightarrow irrelevant(f, g_1, (T, C)) \wedge irrelevant(f, g_2, (T, C))$.

   Proof: From the LHS, we infer that that there is a subset S of T such that $S \vdash_C g_1 \wedge g_2$ and $S \not\vdash_C f$. Therefore, we have $S \vdash_C g_1$ and $S \vdash_C g_2$. Thus, the RHS holds. $\square$.

   To prove this in the reverse direction, we need to make assumptions about the nature of the subsetting construction to show irrelevance. From the RHS, we can infer that there is a subset S_1 of T such that $S_1 \vdash_C g_1$ and $S_1 \not\vdash_C f$, and a subset S_2 of T such that $S_2 \vdash_C g_2$ and $S_2 \not\vdash_C f$. Since S_1 and S_2 are consistent subsets of T that do not entail f, we can construct a subset (not necessarily proper) S of $S_1 \cup S_2$ such that $S \vdash_C g_1 \wedge g_2$ assuming that $g_1 \wedge g_2$ is consistent. The question is whether $S \not\vdash_C f$. If $S_1 \cup S_2$ contains a rule of the form $x_1 \wedge x_2 \Rightarrow f$,

[21] Our implemented irrelevance reasoner uses SLDNF resolution (SLD resolution augmented with negation as failure) which is sound and complete for safe, stratified Horn theories.

where $S_1 \vdash_C x_1$ and $S_2 \vdash_C x_2$, then f is entailed in the union of the two subsets, and not in any of the individual subsets. If our subset construction procedure eliminates all rules that entail f from S_1 and S_2, then $g_1 \wedge g_2$ will still be derivable in the union of S_1 and S_2 without the derivation of f.

4. $irrelevant(f, g_1, (T, C)) \vee irrelevant(f, g_2, (T, C)) \Rightarrow irrelevant(f, (g_1 \vee g_2), ((T, C))$

   Proof: From the RHS we can infer that there is either a subset T_1 of T such that $T_1 \vdash_C g_1$ and $T_1 \not\vdash_C f$, or a subset T_2 of T such that $T_2 \vdash_C g_2$ and $T_2 \not\vdash_C f$. To prove the LHS, we need to establish the existence of a subset S of T such that $S \vdash_C g_1 \vee g_2$ and $S \not\vdash_C f$. We can exhibit $S = T_1$ or $S = T_2$ depending on which disjunct in the LHS was true.

Proposition 19 S1 *satisfies Definition 7.*

Proof: Let $T \vdash_C G$. To make the LHS true, we add the set $derivable\_from(P, (T, C))$ to T. The subset $T^{'}$ of T is constructed by eliminating all clauses in T which contain a positive (non-negated) occurrence of a literal that unifies with P. There are two cases to consider: (1) No rule containing a literal that unifies with P is used in the derivation of G in T. Thus their elimination in the $T^{'}$ construction does not affect the provability of G in $T^{'}$. (2) One of the eliminated rules r is used in a proof of G in $T^{'}$. If P occurs in r's consequent, then the elimination of the rule affects us only if $P \equiv G$. This case is excluded by the antecedent of the schema. If P occurs in r's antecedent, then the proof of G requires the use of the consequent Q of r. By construction, Q occurs in $derivable\_from(P, (T, C))$ and is therefore present in T. So we can reconstruct the proof of G in T using r by clipping it at Q, instead of reducing Q to a set of subgoals including P. $\square$.

Appendix III

Extensional reduction

An incremental procedure for augmentation (shown in Table 5) is implemented in our reduction system. The top-level functions for reduction are shown below. The last rule for `derives` is specific to the kinship example and is automatically generated by analyzing the rules. It is a formulation of **IC1**. The predicate `solves` incrementally solves all goals that use a fact class f for support, and then eliminates f. The reduction process terminates when all irrelevancies are removed. If we run the conditional irrelevance claim reduction scheme shown above on F_1, we extensionally derive a formulation in which `parent` is eliminated and is replaced by the transitively closed `ancestor` relation. This is achieved by a series of `add` and `del` operations (`ck_assert` and `remove` are Prolog implementation of these operations).

```
do_it_all(Dist) :-
  init, iterate_search(Dist).

iterate_search(Dist) :-
   step_search(Dist), iterate_search(Dist).
iterate_search(_).
```

```
reduce(Goal) :- irr(Fact,Goal), fail.
reduce(Goal).

irr(Fact,Goal) :-
   derives(Fact,P),
   setof(_,solves(Fact,P),_),
   remove(Fact),
   fail.
irr(_,_).

solves(Fact,[]).

solves(Fact,[[Prop,Support]|B]) :-
call((Fact,Support)),
ck-assert(Prop),
solves(Fact,B).

derives(parent(X,Y),[[ancestor(X,Z),ancestor(Y,Z)],[ancestor(X,Y),true]]).
```

Table 5: A Prolog implementation of an incremental extensional reducer of conditional irrelevance claims that are generated by the schema in Section 4.2.1.

```
step_search(Dist) :-
  best_del_add(Add,Del,TimeImp,Dist),
  retract(Del), ck_assert_list(Add), add_history(h(Add,Del)).

init:- abolish(history,1), assert(history([])).

best_del_add(Add,Del,Time,Dist) :-
init_best,
generate_del_add,
gather_statistics(Dist),
get_best.

get_best :-
values(Add,Del,T),
if_then_else(mem_history(h(Add,Del)),true,
   (best(_,_,T1),
            if_then_else(T > T1,
      (retract(best(_,B1,T1)),assert(best(Add,Del,T)),!),true), !)),
fail.
get_best.

mem_history(h(A,B)) :- history(H), mem_history(h(A,B),H), !.
mem_history(h(A,B),[h(A,B)|_]) :- !.
```

```
mem_history(h(A,B),[_|H]) :- mem_history(h(A,B),H), !.

gen_del_add :-
possible_del_add(Add,Del),
ck_assert(del_add(Add,Del)),
fail.
gen_del_add.

%% possible del-add is instantiated with an example from Gridworld
%% for expositional convenience.

possible_del_add(conn(X,Y),Sfacts) :-
conn(X,Y), conn(Y,Z),
setof(P,conn(Y,P),S),
make_add_facts(X,S,Sfacts).

make_add_facts(_,[],[]) :- !.
make_add_facts(X,[Y|Rest],[conn(X,Y)|Rest1]) :-
X \== Y,
make_add_facts(X,Rest,Rest1),!.
make_add_facts(X,[Y|Rest],Rest1) :-
make_add_facts(X,Rest,Rest1),!.

gather_statistics(N) :-
del_add(A,B),
implement_del_add(Add,Del,Time,Dist),
assert(values(Add,Del,Time,Dist)),
fail.
gather_statistics(_).

%% effect of implementing a del add.

implement_del_add(Add,Del,Time,Dist) :-
statistics(Dist,Time1),
retract(Del),
ck_asst_list(Add),
statistics(Dist,Time2),
Time is Time1 - Time2,
reverse_del_add(Add,Del),!.

%% undo the effects of a del-add

reverse_del_add(A,B) :-
ck_assert(A),
ck_ret_list(B).
```

Formulation of grid world

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