Finding Optimal Derivation Strategies in Redundant Knowledge Bases

Russell Greiner

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

August 14, 1990

Abstract

A backward chaining process uses a collection of rules to reduce a given goal to a sequence of data-base retrievals. A "derivation strategy" is an ordering on these steps, specifying when to use each rule and when to perform each retrieval. Given the costs of reductions and retrievals, and the *a priori* likelihood that each particular retrieval will succeed, one can compute the *expected cost* of any strategy, for answering a specific query from a given knowledge base. [Smi89] presents an algorithm that finds the minimal cost strategy in time (essentially) linear in the number of rules, for any disjunctive, irredundant knowledge base. This paper proves that the addition of redundancies renders this task NP-hard. Many Explanation-Based Learning systems work by adding in redundancies; this shows the complexities inherent in their task.

1 Introduction

Problem solving is combinatorially expensive. There can be a combinatorial number of potential "solution paths" for a given query — as there can be many rules (a.k.a. operators) that each reduce the goal to a new set of subgoals, and each of these subgoals can, itself, have many possible reductions, etc.; ultimately "bottoming out" with a sequence of data base retrievals. (See [GN87].)

There are several ways of addressing this combinatorial complexity. One involves ordering the set of rules, so the first rule selected for a given (sub)goal is the one viewed as most cost effective. For example, given the K_1 knowledge base, whose rules appear in Figure 1, we may specify that the rule R_{pm} should be used before the rule R_{pf} when determining `Abe`'s parent — *i.e.*, when seeking an x such that `Parent(Abe` x`)` holds. There can, of course, be many comprehensive "derivation strategies" — *i.e.*, many ways of searching this knowledge base, each guaranteed to find a solution, if one exists. Our objective is to find the one that requires the least expected time.

R_{gp}:	`Parent(p q)`	$\Rightarrow$	`Guardian(p q)`
R_{pf}:	`Father(p q)`	$\Rightarrow$	`Parent(p q)`
R_{pm}:	`Mother(p q)`	$\Rightarrow$	`Parent(p q)`

Figure 1: Rules associated with K_1

Another approach involves adding redundant information to the knowledge base, in the form of a new rule. Using K_1 again, we observe that the solution to the `Guardian(Abe Bob)` query involved the fact `Father(Abe Bob)` and the rules R_{gp} and R_{pf}. This suggests modifying our derivation system for subsequent queries: When asked to prove `Guardian(`$\kappa\ \gamma$`)` (for any κ and γ), this "smarter" system will immediately perform the data base retrieval of `Father(`$\kappa\ \gamma$`)`, and only if this fails, consider the other possible retrievals and rule-based reductions (e.g., using R_{gp}, R_{pm}, etc.). This corresponds to combining the rules R_{gp} and R_{pf} to produce the new redundant rule

$$R_{gf}\text{:}\quad \texttt{Father(p q)} \Rightarrow \texttt{Guardian(p q)}$$

that is then incorporated into K_1's set of rules, forming $K_2 \leftarrow K_1 \cup \{R_{gf}\}$. Furthermore, this rule is placed first: the "improved" system will try this new rule first in subsequent queries, before the other rules are attempted. This is the basis for the recent Explanation-Based Learning (EBL) systems [MKKC86, DM86], as well as Chunking [Ros83, RN82, LNR86] and MacroOps [FHN72].

A major objective of these learning systems is *efficiency*: to improve the overall future performance of the system. Of course, this requires some information about these anticipated future events — especially about which questions will be posed and with what distributions, and about the probability that certain assertions will be in the knowledge base when those queries occur.

Many systems implicitly employ the "obvious" assumption that "the future will mirror the past" — that the future questions will correspond to the questions asked until now. This suggests preserving *every* observed rule-sequence as a new redundant rule. Recent empirical evidence [Min85, Min88], however, has confirmed some of the obvious problems inherent in this "save all redundant rules" approach: these new rules can *slow down* the overall performance of the complete system. That is, it is not always advantageous to incorporate a proposed redundant rule into an existing knowledge base.

This report addresses the issue of deciding whether to add in a new redundant rule. It assumes, as given, the *a priori* likelihood that any given database retrieval will succeed. (We may know, for example, that there is a 10% chance that the database retrieval "`Father(`$\kappa\ \gamma$`)`" will succeed, for any pair of constants, $\langle\kappa\ \gamma\rangle$.) It shows how to use this likelihood information to determine both whether a new rule should be added; and if so, where in the derivation strategy that rule should appear.

Section 2 provides a formal definition of "derivation strategies", shows there can be an exponential number of them, and provides a framework for comparing different ones. It also discusses Smith's result [Smi89], which states that this task is (essentially) linear in the number of rules for a certain class of irredundant knowledge bases. Section 3 then addresses the complexities inherent in producing the optimal such strategy when the knowledge base is redundant: It proves first that this task can be achieved efficiently for a certain classes of redundant knowledge bases (*viz.*, knowledge bases produced by some EBL systems) and then that this task is, in general, NP-complete. Section 4 ties this work back into the growing field of EBL systems, and discusses some extensions to this work.

2 Framework

Definition of "Derivation Strategy": Given a specific query, σ (a positive first-order literal), and knowledge base (a collection of first-order definite clauses, containing both rules and ground atomic facts), we define a *(derivational) strategy* as an ordering that specifies when to follow which rules (to reduce the subgoal) and when to perform database lookups. For example, one strategy for answering the query "`Guardian(Abe Bob)`" from K_1 would be

- Lookup `Guardian(Abe Bob)` from (the set of facts in) K_1. If that succeeds, the derivation process returns "`Yes`" and is done. Otherwise:
- Use R_{gp} to reduce this goal to `Parent(Abe Bob)`.
- Lookup `Parent(Abe Bob)` from K_1. If that succeeds, it returns "`Yes`" and is done. Otherwise:
- Use R_{pf} to reduce this subgoal to `Father(Abe Bob)`.
- Lookup `Father(Abe Bob)` from K_1. If that succeeds, it returns "`Yes`" and is done. Otherwise:
- Use R_{pm} to reduce the `Parent(Abe Bob)` subgoal to `Mother(Abe Bob)`.
- Lookup `Mother(Abe Bob)` from K_1.
  If that succeeds, it returns "`Yes`"; otherwise, it returns "`No`". (Either way, it is now done.)

We write this strategy as $\Theta_1 = \langle L_g\ R_{gp}\ L_p\ R_{pf}\ L_f\ R_{pm}\ L_m \rangle$, where R_{xy} represents the reduction using the R_{xy} rule[1], and the L_y steps refer to lookups of the y-related propositions. We refer to R_{xy} steps as "reductions", and to L_y steps as "lookups"; collectively, these are called "steps".

We can use this same strategy, *mutatis mutandis*, to address any query of the form "`Guardian(`$\kappa\ \gamma$`)`". While this approach holds for any arbitrary κ and γ, we will focus on the situation where each is some (unspecified) constant, as opposed to an existentially quantified variable. In all cases, we are seeking *one* answer to the query, rather than *all* solutions. (Hence, the question "`Parent(Abe` x`)`" would seek *one* parent of `Abe`, rather than all of his parents.) This is, therefore, a *satisficing search*, using [SK75]'s notation.

Expected Cost of Derivation Strategy: The *expected cost* of a strategy is the weighted sum of the expected number of lookups plus the expected number of reductions. We assume that each lookup costs ℓ cost-units, and each reduction step, r cost-units. Of course, the expected cost of following a strategy depends critically on the anticipated successes of the lookups, which in turn depend on which facts appear in the knowledge base. If all of Θ_1's lookups fail, this overall strategy will require $3r + 4\ell$ steps. The *expected cost* of a strategy depends on the expected probability of success of each lookup step.[2] Assume, for example, that there is a 0% chance that L_p will succeed (*i.e.*, there are no facts of the form "`Parent(`$\kappa\ \gamma$`)`" in K_1; written "$Pr(L_p) = 0$"), and a 1%, 10% and 25% chance that L_g, L_f and L_m, respectively, will succeed, and that these probabilities are independent. Then the expected cost for this strategy is

$$\begin{aligned} E[\Theta_1] &= \ell + (1 - Pr(L_g))[r + \ell + (1 - Pr(L_p))[r + \ell + (1 - Pr(L_f))[r + \ell + (1 - Pr(L_m))0]]] \\ &= \ell + (1 - \ 0.01\ )[r + \ell + (1 - \ 0\ )[r + \ell + (1 - \ 0.10\ )[r + \ell + (1 - \ 0.25\ )0]]] \\ &= 2.871r + 3.871\ell \end{aligned}$$

[1] In general, a rule may appear many times in a given derivation tree. We would then need to distinguish the different appearances of each rule.

[2] (1) The expected cost can depend on other factors as well, including the probability that a rule may fail to reduce a goal to a subgoal, due to constants appearing in the rules; see [Lik88] and [Smi89]. (2) This note assumes that we know, *a priori*, the probability that each retrieval will succeed, and is not concerned with how they were obtained. [Smi89, Lik88, Cho90] present one way of estimating these values, based on the distribution of assertions present in the knowledge base; [GO90] presents a different model.

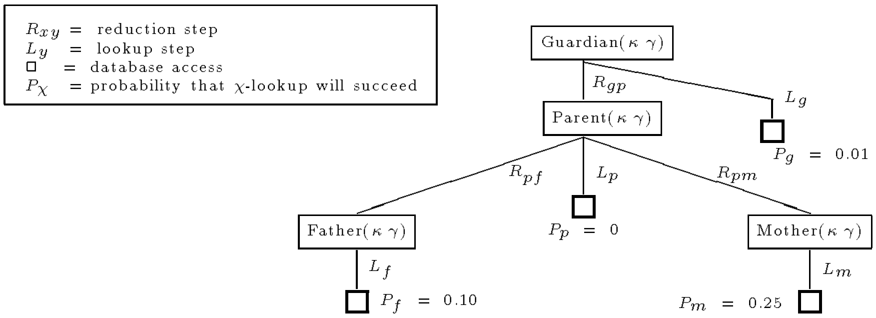

Figure 2: Inference Graph of K_1's Rules

There can, of course, be many strategies for a given goal within a given knowledge base. One could, for example, not bother trying to retrieve `Parent(···)` from K_1, and also follow the R_{pm} rule and its associated `Mother(···)` lookup before R_{pf} and `Father(···)`. The expected cost of this alternative strategy, $\Theta_2 = \langle L_g\ R_{gp}\ R_{pm}\ L_m\ R_{pf}\ L_f \rangle$, is $E[\Theta_2] = \ell + (1-0.01)[r + r + \ell + (1 - 0.25)[r + \ell + (1-0.10)0]] = 2.7225r + 2.7325\ell$, which is strictly less than $E[\Theta_1]$ for any values of r and ℓ.

Nothing forces us to consider lookups before reductions. The $\Theta_3 = \langle R_{gp}\ R_{pm}\ L_m\ R_{pf}\ L_f\ L_g \rangle$ strategy, for example, does not bother to perform the (low probability) `Guardian(···)` lookup until the end. Its cost $E[\Theta_3] = 2.75r + 2.425\ell$ can be yet less expensive — e.g., if $r = 1$ and $\ell = 2$, then this Θ_3 is the least expensive strategy yet, as $E[\Theta_3] = 7.6 < 8.1685 = E[\Theta_2]$.

This paper deals only with *complete* and *minimal* derivation strategies, where

Definition 2.1 *A strategy is* complete *if it is guaranteed to find an answer, if there is one.*
A strategy is minimal *if no proper initial subsequence has the same cost.*

All three strategies shown above (Θ_1, Θ_2 and Θ_3) are complete. Notice that a strategy can ignore the L_p lookup step and still be complete, as $Pr(L_p) = 0$.
Every strategy is minimal unless it includes a retrieval step that has a 100% chance of succeeding. For example, observe that the expected cost of $\langle L_q \cdots \rangle$ is r if $Pr(L_q) = 1$, independent of what steps follow L_q. Hence, $\Theta_{Q1} = \langle L_q, R_{p \Rightarrow q}, L_p \rangle$ is not minimal, as its cost, $E[\Theta_{Q1}]$, is r, as is a proper subsequence, $\Theta_{Q2} = \langle L_q \rangle$; this Θ_{Q2} sequence, however, is minimal.

Definition 2.2 *An* optimal derivation strategy*, for a given query from a given knowledge base, is a complete minimal strategy with the least expected cost.*

Notice there can be more than one such optimal strategy.

We can now state our objective: to describe the complexities inherent in computing these optimal derivation strategies, given various different classes of knowledge bases.

Number of Derivation Strategies: As shown above, a derivation strategy, for a given query from a knowledge base K, is an ordered sequence of a subset of K's reduction and retrieval steps.

If there are n such steps, the number of possible (not necessarily minimal nor complete) strategies is $\sum_{i=0}^{n} \binom{n}{i} \times i!$ (as there are $\binom{n}{i} \times i!$ different strategies of length exactly i), which is exponential in n. Even for the tiny K_1 knowledge base, with $n = 7$, there are 13,700 different strategies, of which $6! = 720$ are complete and minimal, and of these, 36 respect the "subgoal precedence order" (*e.g.*, only perform the L_f step after R_{pf}, etc.).

Smith [Smi89], however, shows how to compute the optimal derivation strategy in a time (essentially) proportional to the number of rules, for any *disjunctive*, *function-free*, *fact-irredundant*, *rule-irredundant* knowledge base.[3] *Disjunctive* means that all of the rules are of the form $\mathtt{A}(\cdot) \Rightarrow \mathtt{C}(\cdot)$; hence, it excludes rules of the form $\mathtt{A}(\cdot)\&\mathtt{B}(\cdot) \Rightarrow \mathtt{C}(\cdot)$. (Stated formally, it includes only definite clauses with exactly one positive literal, and either zero or one negative literals.) *Function-free* means it does not allow embedded function symbols. *Fact-irredundant* means that no existing database ground unit clause can be derived from the knowledge base of rules and the other data. (Hence, if the knowledge base includes the R_{pf} rule, it may not contain both `Father(A B)` and `Parent(A B)`.) *Rule-irredundant* means that there is at most one derivation path that connects any goal with any of its subgoals; see Definition 3.2. This means that the inference graph — the graph whose nodes are (sub)goals and whose arcs represent the rules that link a goal to its children; see Figure 2 — is a tree, rather than a more general directed acyclic graph ("dag").

N.b., this note deals only disjunctive, function-free, fact-irredundant knowledge bases; and contrasts rule-irredundant and rule-redundant variants. To simplify notation, we will use the term "redundant" to mean "rule-redundant".

3 Dealing with Redundant Knowledge Bases

Many knowledge bases, however, are *redundant* — that is, include two paths joining a pair of nodes (see Definition 3.2). One reason is that few (if any) real systems prevent the user from entering redundancies. (In general, the test to determine whether a set of first-order statements is redundant is undecidable, as it involves determining whether $K - R \models R$, for each member $R \in K$.) Another reason was mentioned above: many "explanation-based learning" systems automatically add in redundant rules.

The rest of this note focuses on such knowledge bases. The K_2 knowledge base is an example. It is redundant because its inference graph (shown in Figure 3) includes two paths that join the query `Guardian(`$\kappa\ \gamma$`)` to the `Father(`$\kappa\ \gamma$`)` subgoal — one using $\langle R_{gp}\ R_{pf} \rangle$, and the other, $\langle R_{gf} \rangle$.

This section addresses the complexities inherent in finding optimal derivation strategies for redundant knowledge bases. Subsection 3.1 first presents an important reduction. Subsection 3.2 uses it to characterize a class of redundant knowledge bases for which this task is polynomial in the size of the inference graph. Section 3.3 proves, however, that the task is NP-complete in its full generality (even for disjunctive, function-free, data-irredundant knowledge bases).

We first provide some essential definitions and simplifications:

Definition 3.1 *A* path *from the goal G to one of its subgoals S is a sequence of rules $\langle r_1, \ldots, r_k \rangle$ such that the conclusion of r_1 matches G, the antecedent of each r_i matches the conclusion of r_{i+1}*

[3](1) Actually, Smith's approach extends beyond these knowledge bases. (2) That algorithm involves sorting the set of m steps that can apply at each subgoal — hence requiring an additional factor of $O(m \log(m))$. In general, though, $m \ll$ the number of steps.

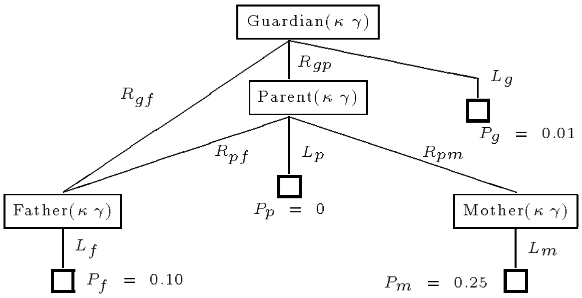

Figure 3: Inference Graph of K_2's Rules

(for $i = 1..k-1$), and the antecedent of r_k matches S.
We say this path "leads from S up to G".

Definition 3.2 *A knowledge base is* redundant *iff it includes any pair of (sub)goals, $\langle G, S\rangle$, such that there is more than one path leading from S up to G.*

To simplify our analysis, we will assume that rules include no object constants — that is, all arguments of all relations that appears in all rules are variables. Hence, we allow all of the rules shown above — R_{gp}, R_{pm}, R_{pf}, R_{gf} — but exclude rules like R_{bad} : `Father(x Mark)` $\Rightarrow$ `Guardian(x Mark)`. This allows us to deal exclusively with the rule's relations, as we know that $S_1(\cdots) \Rightarrow S_2(\cdots)$ can be used to reduce any goal of the form $S_2(\cdots)$. It is easy to extend this analysis to cover this more general case in which rules may include constants; see [Smi89] and [Lik88].

3.1 Irredundant Derivation Subspaces

A derivation strategy is *redundant$_S$* iff it includes the same step more than once; *e.g.*, $\Theta_4 = \langle R_{gf}\ \underline{L_f}\ R_{gp}\ R_{pm}\ L_m\ R_{pf}\ \underline{L_f}\ L_g\rangle$ is redundant$_S$ as it includes L_f (*i.e.*, asks for a `Father(·, ·)` proposition) twice. It obviously *never* makes sense to use a redundant$_S$ derivation strategy to solve a specific query, as there is always an irredundant$_S$ strategy that is functionally equivalent (*i.e.*, will find an answer whenever the redundant$_S$ strategy does) and that takes strictly less time. (See [Gre89, Lemma 3.1].) Here, $\Theta_5 = \langle R_{gf}\ L_f\ R_{gp}\ R_{pm}\ L_m\ L_g\rangle$ is such a reduced, irredundant$_S$ strategy for Θ_4. (Notice Θ_5 is the subsequence of Θ_4 that omits both the second L_f lookup and the now no-longer-useful R_{pf} reduction step.)

This means we need only consider *irredundant derivation subspaces*. That is, let $RS(\Theta)$ map the strategy Θ into the set of rules it uses — *e.g.*, $RS(\Theta_3) = \{R_{gp}, R_{pm}, R_{pf}\}$. Notice the rule set of an irredundant$_S$ strategy corresponds to an irredundant knowledge base, which means we can use [Smi89]'s algorithm to find the optimal strategy in linear time.[4] Hence, we can reduce the problem of finding the "optimal derivation strategy" to the problem of finding the "optimal derivation *space*", where each "derivation space" is the rule set plus the needed lookup steps.

Unfortunately, there can be an exponential number of derivation spaces. Consider, for example, the "trellis" inference graph, formed from the rules

$$K_3 \quad = \quad \{\, \mathtt{A}_i(x) \Rightarrow \mathtt{A}_{i+1}(x)\,\}_{i=1}^{n+1} \quad \bigcup \quad \{\, \mathtt{A}_i(x) \Rightarrow \mathtt{A}_{i+2}(x)\,\}_{i=1}^{n}$$

[4] Recall we are assuming that the knowledge base is disjunctive, function-free and data-irredundant.

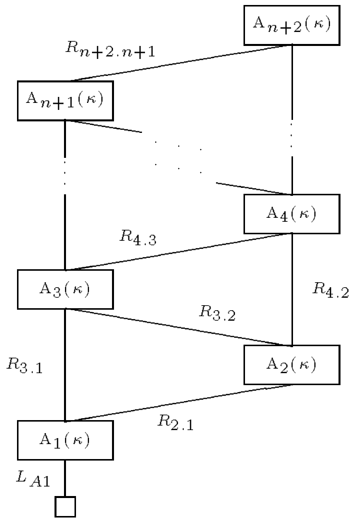

Figure 4: "Trellis Inference Graph", for K_3

shown in Figure 4. It has $Fib(n)$ irredundant derivation spaces, each connecting the goal $\mathtt{A}_{n+2}(\kappa)$ to the L_{A1} lookup. ("$Fib(m)$" is m^{th} Fibonacci number.)

3.2 Simple Redundant Knowledge Bases

Fortunately, there are efficient ways of selecting the optimal space, for certain classes of knowledge bases. All that is required is a polynomial time algorithm that can select, for each subgoal S, which step to follow "up" — that is, which selects one step from the set of all steps beginning paths that lead from S up to the top level goal G. (*E.g.*, given the K_2 inference graph of rules, this algorithm would have to specify whether to take the R_{pf} or R_{gf} reduction from the `Father(`$\kappa\ \gamma$`)` subgoal, to lead up to `Guardian(`$\kappa\ \gamma$`).`) In fact, this condition is both necessary and sufficient to guarantee that an efficient "find optimal derivation strategy" process will exist for a knowledge base. Hence, we can reduce this task to one of finding the optimal "upward" step at each node.

This subsection proves that this "which step to follow" decision is trivial for the class of redundant knowledge bases produced by some EBL systems, meaning that the "find optimal derivation strategy" task is polynomial in the size of their inference graphs.[5] Subsection 4.1 discusses some limitations of this result.

We begin with some definitions:

Definition 3.3 *A* σ-direct rule *is a rule whose conclusion matches* σ*.*

E.g., R_{gf} and R_{pf} are each "`Father(`$\cdot$`)`-direct rules" in K_2.

Definition 3.4 *The path* $\rho_1 = \langle r_1 \cdots r_k \rangle$ *is "*redundant$_P$*" with the path* $\rho_2 = \langle s_1 \cdots s_l \rangle$ *iff the conclusion of* r_1 *matches the conclusion of* s_1 *and the antecedent of* r_k *matches the antecedent of* s_l *(i.e., both paths lead from the same subgoal, and arrive at the same subgoal), and they share no*

[5] The extended paper, [Gre89], presents yet other classes of knowledge bases in which we can efficiently find optimal derivation strategies.

*steps in common (*i.e., $\{r_i\}_i \cap \{s_j\}_j = \{\}$*).*
*Here, we say that the final steps, r_k and s_l, are "*redundant$_R$ (wrt the r_1's conclusion)*" as their antecedents match and they each begin paths leading to the same conclusion.*
*The set of paths $\{\rho_i\}_{i=1}^{r}$ is a "*redundant$_P$ path set*" iff $r > 1$ and for each $\rho_m, \rho_n \in \{\rho_i\}_i$, ρ_m is redundant$_P$ with ρ_n whenever $\rho_m \neq \rho_n$; and this set is a "*maximal redundant$_P$ path set*" if it includes every path redundant$_P$ with ρ_1.*
We can say that a (maximal) redundant$_P$ path set "leads from γ to σ" iff each member path leads from γ to σ.

E.g., for the K_2 inference graph: the paths $\langle R_{gp}\ R_{pf} \rangle$ and $\langle R_{gf} \rangle$ are redundant$_P$ as each leads from `Father(`$\kappa\ \gamma$`)` to `Guardian(`$\kappa\ \gamma$`)`; and the set $\{\langle R_{gp}\ R_{pf}\rangle,\ \langle R_{gf}\rangle\}$ is a maximal redundant$_P$ path set. Finally, R_{pf} and R_{gf} are redundant$_R$ (with respect to `Guardian`).

Definition 3.5 *The knowledge base K is a "*σ-direct-KB*" if every redundant path set leads (from some goal) to the goal σ, and each maximal redundant path set includes a σ-direct rule (*i.e.*, a single rule whose antecedent is the antecedent of the rule set, and whose conclusion is the goal σ).*

As examples: K_1 is a "`Guardian`-direct-KB" vacuously, as it includes no redundant$_P$ path sets; K_2 qualifies, as its only redundant$_P$ path set includes the R_{gf} rule which leads directly to `Guardian(`$\kappa\ \gamma$`)`; but K_3 (Figure 4) is not a "$\mathtt{A}_{n+2}$-direct-KB" (for $n > 1$) as it includes the $\{\ \langle R_{3.2},\ R_{2.1}\rangle,\ \langle R_{3.1}\rangle\ \}$ maximally redundant$_P$ path set that does not include a $\mathtt{A}_{n+2}$-direct-rule.

Let Θ be the optimal derivation strategy associated with the top-level goal, σ, within the irredundant disjunctive knowledge base, K. Assume K includes a path from σ to a subordinate subgoal, λ, of the form $\langle R_{\sigma 2}\ R_{23}\ \cdots\ R_{n\lambda}\rangle$, whose final step, $R_{n\lambda}$, represents the rule "$\lambda \Rightarrow S_n$". Now let $R_{\sigma\lambda}$ be an additional σ-direct rule (not in the initial K) that immediately connects σ with λ — *i.e.*, of the form "$\lambda \Rightarrow \sigma$". (This is the type of rule that most EBL processes would generate, after solving σ. The R_{gf} rule is an example, as it directly connects the `Guardian(`$\cdots$`)` query with the `Father(`$\cdots$`)` subgoal.) Now form a new knowledge base by replacing the final rule used in the original σ to λ path (here, $R_{n\lambda}$), by the σ-direct rule (here, $R_{\sigma\lambda}$). The resulting set, $RS(\Theta) - \{\ R_{n\lambda}\ \} + \{\ R_{\sigma\lambda}\ \}$ is an irredundant derivation space; let Θ' be the optimal strategy for this space. Lemma 3.1, below, proves that $E[\Theta'] \leq E[\Theta]$, which means that an optimal derivation strategy (and therefore, the optimal derivation space) will include the direct rule. (This is good news for EBL fans, as these direct rules are exactly what EBL systems generate! ...but see the comments in Subsection 4.1.)

Hence, there is an obvious linear time algorithm for finding an optimal derivation strategy for σ, for the simple case of adding a new σ-direct redundant rule to a previously irredundant (disjunctive, function-free, data-irredundant) knowledge base: Add the new direct rule, and remove the rule with which it was redundand$_R$.[6] Then use [Smi89]'s algorithm to produce the optimal strategy for this new knowledge base.

A straightforward extension allows us to find, in linear time, the optimal strategy for the σ query from any "σ-direct-KB", defined in Definition 3.5. An optimal strategy for σ from a σ-direct-KB can be found in the derivation space that includes only the σ-direct rule in each redundant$_P$ path

[6] We assume that our strategy-finding system knows which arcs are redundant. This is quite reasonable within the EBL context.

set, and excludes all of the rules in the other members of each redundant$_P$ path set; see Lemma 3.2. This means we can immediately determine which rule to follow up from each subgoal, leading to an efficient algorithm for this class of knowledge bases.

Lemma 3.1 (from [Lik88, Lemma 3.2]) *Let K be an irredundant knowledge base that includes a path of rules from some query, σ, to the subgoal, λ, whose final step involves the rule $R_{n\lambda}:\ \lambda \Rightarrow S_n$. Let Θ_1 be the optimal strategy for σ from K. Let K' be identical to K* EXCEPT *it replaces $R_{n\lambda}$ with a new σ-direct rule, $R_{\sigma\lambda}$ — that is "$\lambda \Rightarrow \sigma$"; and let Θ_2 be the optimal strategy for σ from K'. Then $E[\Theta_2] \leq E[\Theta_1]$.*

Proof: It suffices to show that there is *some* complete strategy for deriving σ from K' whose cost is no greater than $E[\Theta_1]$. (This K' strategy need not be the optimal one.) There are two cases, depending on whether $R_{n\lambda}$ appears in Θ_1.
(1) If $R_{n\lambda}$ is not in Θ_1, then Θ_1 also qualifies as a K' strategy. It is clearly still complete; and its cost, obviously, is $\leq E[\Theta_1]$.
(2) If $R_{n\lambda}$ is in Θ_1, define Θ_3 as the strategy that is identical to Θ_1, except it replaces the single occurrence of $R_{n\lambda}$ with $R_{\sigma\lambda}$. It is trivial to show that Θ_3 is a complete strategy for deriving σ from K', as Θ_1 is; and that $E[\Theta_3] = E[\Theta_1]$. □

Lemma 3.2 (from [Lik88, Theorem 5.1]) *Let K be a σ-direct-KB, whose maximal redundant$_P$ path sets are $\{\tau^i\}_{i=1}^k$, where each $\tau^i = \{\gamma_j^i\}_{j=1}^{n_i}$ contains γ_1^i which is a σ-direct rule (i.e., γ_1^i leads directly to σ). Then there is an optimal strategy for σ from K that does not involve any of the $\{\gamma_j^i\}_{j=2}^{n_i}$ for any i.*

Proof: Consider first the case where $k = 1$ — *i.e.*, where there is but a single redundant$_P$ path set, $\tau^1 = \{\gamma_j^1\}_{j=1}^{n_1}$. Lemma 3.1 handles the case where this path set includes only two paths (ie $n_1 = |\tau^1| = 2$); trivial induction on this size of this $|\tau^1|$ deals with other redundant$_P$ paths (*i.e.*, allows the lemma to hold for arbitrarily sized τ_1). Using this $k = 1$ as the base case, we can inductively show that this property holds for any number of redundant$_P$ path sets (*i.e.*, for any k). □

3.3 Finding the Optimal Strategy is NP-Complete

This subsection proves that the task of finding the optimal derivation strategy for a given query from an arbitrary redundant knowledge base is NP-complete. The basic problem, as suggested in Subsection 3.1, is that the optimal strategy can involve the steps from any of an exponential number of possible irredundant subspaces; and there appears to be no simple way of determining which space it is. In particular, there is no efficient (*i.e.*, polynomial time) process that can determine which "upward" path to follow from any given subgoal for arbitrary inference graphs (assuming $P \neq NP$).

To state this formally, first observe that if we can find the optimal derivation strategy in polynomial time, then we can solve (in polynomial time) the related decision problem, *OptDS*:

Definition 3.6 (*OptDS* Decision Problem)
INSTANCE: *A knowledge base K, a query σ, a probability function $Pr(\cdot)$ that maps each retrieval L_i to its probability of success $Pr(L_i)$, values for the cost of each reduction step r and data-base*

retrieval ℓ, and a constant k.
QUESTION: *Does there exist a complete, minimal, irredundant derivation strategy for answering σ from K whose expected cost is less than or equal to k — i.e., is there a Θ such that $E[\Theta] \leq k$?*

Now for the main theorem:

Theorem 3.1 *The OptDS task is NP-complete.*

Proof: To show that OptDS is in NP: Given a redundant knowledge base with a total of n steps (including both rules and retrievals), we need only guess an appropriate sequence of a subset of these steps from the set of $\sum_{i=0}^{n} \binom{n}{i} \times i! \;\leq\; e \times n! \;=\; O(n^n)$ such ordered subsequences, and confirm (in polynomial time) that it is a complete, minimal, irredundant strategy whose expected cost is $\leq k$.

To show that OptDS is NP-hard, we reduce the NP-complete "Exact Covering" task [GJ79] to it.[7]

Definition 3.7 (*EC* Decision Problem)
INSTANCE: *A collection of sets, $\mathcal{C} \;=\; \{\, c_i \,\}_i$. Let $\mathcal{S}$ be the union of its members — $\mathcal{S} \;=\; \bigcup_{c_i \in \mathcal{C}} c_i$.*
QUESTION: *Does there exist a subset $\mathcal{C}' \subseteq \mathcal{C}$ such that the members of $\mathcal{C}'$ are pairwise disjoint and exhaust $\mathcal{S}$ — i.e., $\forall c_i, c_j \in \mathcal{C}'\; c_i \neq c_j \Rightarrow c_i \cap c_j = \{\}$ and $\bigcup_{c_i \in \mathcal{C}'} c_i \;=\; \mathcal{S}$?*

Given any such collection $\mathcal{C}$, we can form a knowledge base as follows: Associate each $c_i \in \mathcal{C}$ with a proposition (a.k.a. "derivation space node") $\mathtt{C}i\mathtt{(x)}$; likewise identify each $s_k \in \mathcal{S}$ with the proposition $\mathtt{S}k\mathtt{(x)}$. The only retrievals with non-zero probabilities of success are for these $\mathtt{S}k\mathtt{(x)}$ nodes. Now for the rules: For each $c_i \in C$, of size $n_i \stackrel{def}{=} |c_i|$, add in the n_i rules: $\mathtt{C}i(x) \Rightarrow \mathtt{C}i_2(x)$, $\mathtt{C}i_2(x) \Rightarrow \mathtt{C}i_3(x)$, …, $\mathtt{C}i_{n_i-1}(x) \Rightarrow \mathtt{C}i_{n_i}(x)$ and $\mathtt{C}i_{n_i}(x) \Rightarrow \mathtt{G}(x)$. (These $n_i - 1$ $\mathtt{Ci}_j$ goals are all new, as is the common root, $\mathtt{G(x)}$.) Notice there are exactly n_i reduction steps from $\mathtt{G(x)}$ down to $\mathtt{C}i\mathtt{(x)}$. Now, for each $s_k \in c_i$, add in the additional rule $\mathtt{S}k(x) \Rightarrow \mathtt{C}i(x)$. Define $R \stackrel{def}{=} \sum_i n_i$; observe there are exactly $2 \times R$ rules in this knowledge base — R rules connecting $\mathtt{G(x)}$ to the set of all $\{\mathtt{C}i\mathtt{(x)}\}_i$ nodes, and another R connecting these nodes with the set of all $\{\mathtt{S}k\mathtt{(x)}\}_k$ nodes. Furthermore, define $L \stackrel{def}{=} |\mathcal{S}|$; this is the number of lookups. Then $A \stackrel{def}{=} 2R + L$ is the total number of steps in the entire search space, including both reductions and lookups; it corresponds to the number of arcs in the inference graph, as shown in Figure 5.

Now for the constants: Set $r \;=\; \ell \;=\; 1$, and set the probability of success of each lookup to $Pr(L_i) \;=\; 1 - [1 - \frac{1}{2A}]^{1/L}$. Finally, let $k \;=\; 3L$.

We need only show that there is a derivation strategy for $\mathtt{G(x)}$ from this knowledge base with cost $\leq 3L$ iff there is an exact covering of the sets in $\mathcal{C}$.

We can simplify this equivalence by considering "tree costs": For any possible strategy, Θ, define $T[\Theta]$ to be Θ's "tree cost"; that is, the number of steps (read "arcs") included in Θ. (Notice this sum is not weighted with the probability values.) Lemma 3.3 below shows that $T[\Theta] - \frac{1}{2} \leq E[\Theta] \leq T[\Theta]$ for any strategy, given the above specifications.

This means that a strategy's expected cost is $\leq 3L$ iff its tree cost is $\leq 3L$. (To see this, observe that Lemma 3.3 means that $E[\Theta] \leq 3L$ implies $T[\Theta] \leq E[\Theta] + \frac{1}{2} \leq 3L + \frac{1}{2}$, meaning $T[\Theta] \leq 3L$ as

[7] This proof is an extension of Karp's proof that the "Steiner Tree decision problem" is NP-hard [Kar72].

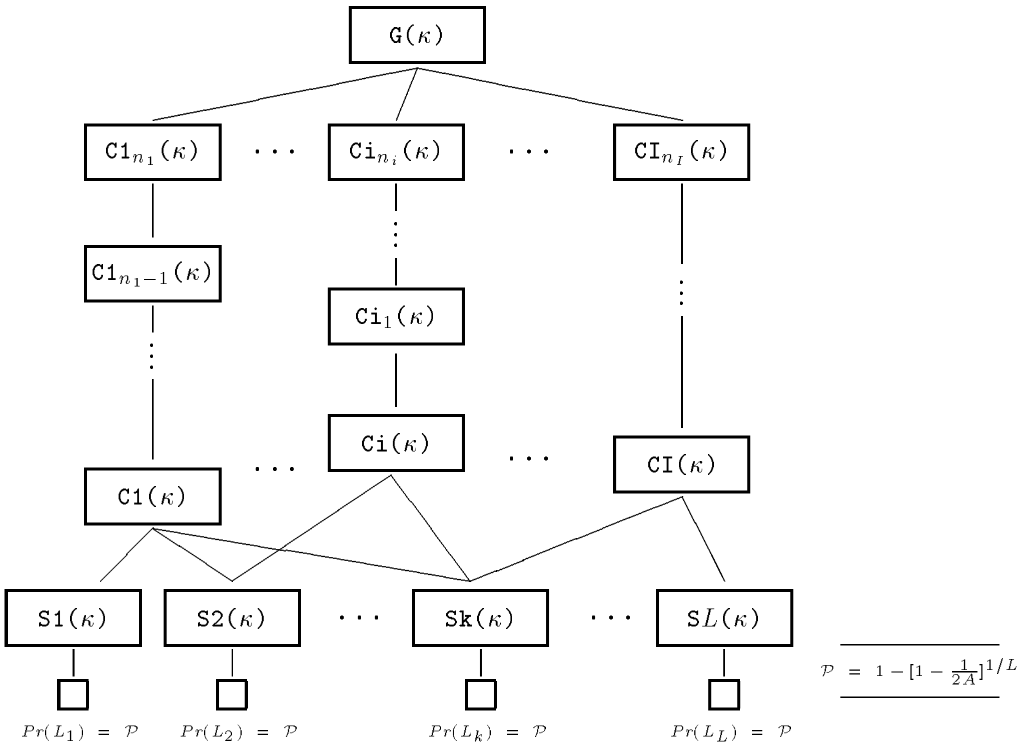

Figure 5: General Inference Graph used in Proof of Theorem 3.1

both $3L$ and $T[\Theta]$ are integers. The other direction is immediate.) Hence, we need only show that there is a strategy whose tree cost is $\leq 3L$ iff there is an exact covering of the sets in $\mathcal{C}$. (That is, we need not consider the order of the strategy's steps.)

$\Longleftarrow$: Assume there is an exact cover of the $\mathcal{C}$ set; w.l.o.g., write this $\mathcal{C}'$ set as $\{c_1, \cdots c_m\}$. Let C_t represent the nodes in the corresponding inference tree — *i.e.*, $C_t = \{\mathtt{C}i\mathtt{(x)}\}_{i=1}^{m}$. Now consider the tree in the derivation space that includes exactly the rules from `G(x)` down to each $\mathtt{C}i\mathtt{(x)}$ in C_t, the rules from each of these $\mathtt{C}i\mathtt{(x)}$s to each of its $\mathtt{S}k\mathtt{(x)}$s, and the lookups under each $\mathtt{S}k\mathtt{(x)}$. Call this subgraph Θ. It suffices to show that Θ is a complete, minimal, irredundant derivation strategy whose tree cost is exactly $3L$. (Technically, we should define Θ to be any legal strategy that involves exactly these steps — that is, any ordering on these steps. However, we saw above that this ordering is factored out.)

Θ's minimality follows from the observation that none of the probabilities are 1. Θ's completeness follows immediately from the claim that the elements of $\mathcal{C}'$ exhaust all of $\mathcal{S}$; its irredundancy follows from the claim that these elements are pairwise disjoint. Θ's tree cost is the sum of (i) the cost of rules from the `G(x)` down to each $\mathtt{C}i\mathtt{(x)}$ in C_t, plus (ii) the cost of the rules from each of these $\mathtt{C}i\mathtt{(x)}$s down to their respective $\mathtt{S}k\mathtt{(x)}$s, plus (iii) the cost of the retrieval step under each $\mathtt{S}k\mathtt{(x)}$. Each portion costs exactly L:
(i) The cost from `G(x)` to $\mathtt{C}i\mathtt{(x)}$ is n_i, hence (i) costs $\sum_{i=1}^{m} n_i$. As $\{c_i\}_{i=1}^{m}$ is an exact cover, we know that $\sum_{i=1}^{m} n_i = \sum_{i=1}^{m} |c_i| = |\cup_{i=1}^{m} c_i| = |\mathcal{S}| = L$.
(ii) There are exactly n_i rules under each $\mathtt{C}i\mathtt{(x)}$; once again, this means this part costs $\sum_{i=1}^{m} n_i = L$.
(iii) As there are $|\mathcal{S}| = L$ $\mathtt{S}k\mathtt{(x)}$ nodes, each appearing in this tree, the cost of the final portion is again L.
Hence, $T[\Theta] = 3L$, meaning $E[\Theta] \leq 3L$.

$\Longrightarrow$: Assume Θ is a minimal complete irredundant strategy whose tree cost is $\leq 3L$; w.l.o.g., assume it uses the nodes $C_t = \{\texttt{C}i\texttt{(x)}\}_{i=1}^m$. It suffices to show that the corresponding set, $\{c_i\}_{i=1}^m$, is an exact cover of $\mathcal{C}$. This translates into the requirement that, for each $\texttt{S}k\texttt{(x)}$ node, there is exactly one "$\texttt{S}k(x) \Rightarrow \texttt{C}i(x)$" rule leading up to a unique $\texttt{C}i\texttt{(x)}$ in C_t.

As Θ is complete, we know there is at least one such rule for each $\texttt{S}k\texttt{(x)}$ node; assume this rule leads to the node $\texttt{C}i_k\texttt{(x)}$. (Notice $\texttt{C}i_k\texttt{(x)}$ is in C_t, by construction.) As Θ is irredundant, we know that Θ can include (no more than) one such $\texttt{C}i\texttt{(x)}$ node. We need to show, however, that there are no other arcs leading from this $\texttt{S}k\texttt{(x)}$ up to any other node in C_t.

Consider the number of arcs connecting $\texttt{G(x)}$ down to each $\texttt{C}i\texttt{(x)}$ node; by construction, this involves n_i reduction steps. Hence, the total number of reduction steps from $\texttt{G(x)}$ to the full set of C_t nodes is $\sum_{i=1}^m n_i$. This sum will be greater than L iff there is any pair, $\langle \texttt{C}i\texttt{(x)},\ \texttt{C}j\texttt{(x)} \rangle$, whose corresponding $\langle c_i,\ c_j \rangle$ contain a common element. Hence, to show that no such pair exists, we need only show that the total cost of the tree from $\texttt{G(x)}$ to the C_t nodes is $\leq L$.

As Θ is irredundant, it includes exactly one "$\texttt{S}k\texttt{(x)} \Rightarrow \texttt{C}i\texttt{(x)}$" rule for each of the L $\texttt{S}k\texttt{(x)}$'s nodes, and exactly one retrieval under each of these L nodes. These parts of the tree account for a cost of $L + L$. As Θ's total tree cost is $\leq 3L$, the total cost associated with reducing $\texttt{G(x)}$ down to these $\texttt{C}i\texttt{(x)}$s must be $\leq 3L - 2L = L$, as desired. $\square$

Lemma 3.3 *Given the conditions shown above, the expected cost of any strategy Θ for $\texttt{G(x)}$ from this knowledge base is between $T[\Theta] - 1/2$ and $T[\Theta]$, where $T[\Theta]$ is Θ's "tree cost".*

Proof: First, express any strategy, Θ, as a sequence of reduction chains, $\langle r_1\ r_2\ \cdots\ r_k \rangle$, where each r_i is a sequence of reductions followed by a lookup step. Notice Θ's tree-cost is $T[\Theta] = \sum_{i=1}^k C(r_i)$, where $C(r_i)$ is the total number of reduction and lookup steps involved in the reduction chain r_i; and Θ's expected cost is

$$E[\Theta] \quad = \quad \sum_{i=1}^{k} \left(C(r_i) \ \times \ \prod_{j=1}^{i-1} [1 - P_j] \right) \tag{1}$$

where P_j is the probability of success of r_j's terminal lookup step. Using P_j's specified value, $1 - [1 - \frac{1}{2A}]^{1/L}$, Equation 1 reduces to

$$E[\Theta] \ = \ \sum_{i=1}^{k} \left(C(r_i) \times \prod_{j=1}^{i-1} 1 - \ [1 - \left[1 - \frac{1}{2A}\right]^{1/L}] \right) \ = \ \sum_{i=1}^{k} \left(C(r_i) \times \left[1 - \frac{1}{2A}\right]^{(i-1)/L} \right) \tag{2}$$

Given Equation 2, using $[1 - \frac{1}{2A}] < 1$ and $i - 1 \geq 0$,

$$E[\Theta] \quad < \quad \sum_{i=1}^k (C(r_i) \times 1) \quad \leq \quad \sum_{i=1}^k C(r_i) \quad \leq \quad T[\Theta]$$

For the other inequality: as $i \leq k \leq L$ and $[1-\frac{1}{2A}] < 1$ and $[1-\frac{1}{2A}]^{(i-1)/L} \geq [1-\frac{1}{2A}]^{k/L} \geq [1-\frac{1}{2A}]^{L/L}$, we can translate Equation 2 into

$$\begin{aligned} E[\Theta] \ &\geq \ \sum_{i=1}^{k} \left(C(r_i) \times [1 - \tfrac{1}{2A}]^{L/L} \right) \ && \geq \ \left(\sum_{i=1}^{k} C(r_i) \right) \times [1 - \tfrac{1}{2A}] \\ &\geq \ T[\Theta] \times [1 - \tfrac{1}{2A}] && \geq \ T[\Theta] - \tfrac{T[\Theta]}{2A} \end{aligned}$$

As Θ uses only a subset of the total set of steps, $A \geq T[\Theta]$, so

$$E[\Theta] \quad \geq \quad T[\Theta] - \tfrac{A}{2A} \quad \geq \quad T[\Theta] - \tfrac{1}{2}$$

as desired. $\square$

4 Conclusions

This concluding section discusses the relevance of this work by tying it into the framework of EBL systems in general, and then lists some obvious extensions to this work.

4.1 Tie to EBL Systems

Lemma 3.1, which proves that a direct redundant rule (*i.e.*, the result of some EBL systems) can never slow down a derivation system, should be viewed as only a partial vindication of these EBL systems and techniques. Below are some comments about this claim:

- This result only holds for disjunctive, function-free, fact-irredundant knowledge bases. In particular, it does not apply if any rule's antecedent includes more than one conjunct. (See [Min88].)
- This result applies only when the initial knowledge base is irredundant. We saw the situation is much more complicated when we consider arbitrarily redundant knowledge bases. Here, it is NP-hard even to determine whether to add in a proposed new rule. (This is equivalent to the task of determining whether an optimal strategy, from a redundant knowledge base, should include a specific rule. If this had a polynomial time solution, then we could solve the *OptDS* task in polynomial time, by first using this test as a filter on the rules, and then using [Smi89]'s linear-time algorithm on the irredundant set of surviving rules.)
- Most of these EBL systems leave in both the direct rule and the rules from which it was derived (*e.g.*, both the derived R_{gf} and the pair R_{gp} and R_{pf}), and will use both sets of rules when solving a single query. We showed above that this is never efficient, even for the motivating query itself.

  (Of course, whenever we want the overall system to be able to solve other queries, which involve other nodes in the inference graph, we must leave both rules in the overall knowledge base, even though this renders it redundant. We are arguing for using some control system that prevents both redundant rules from being used for a single query. The cost of performing this meta-level computation can be quite small.[8])
- Most of these EBL systems move this new rule to the *beginning* of the system's derivation strategy; this is not always the optimal place. Assume that Θ_3 was the optimal strategy for K_1, and consider adding in the new rule R_{gf}: `Father(p q)` $\Rightarrow$ `Guardian(p q)`. If we add it to the front of Θ_3 (and delete R_{pf}), we form $\Theta_6 = \langle R_{gf}\ L_f\ Igp\ R_{pm}\ L_m\ L_g \rangle$, which is a *worse* strategy, as $E[\Theta_6] = 2.8r + 2.575\ell$ is always more than $E[\Theta_3] = 2.75r + 2.425\ell$.

[8] This argues for a reasoning system that can guide its derivation process by selecting which rules to fire at each time — such as MRS [Rus85].

(The optimal ordering for this expanded knowledge base, by the way, is $\Theta_{6a} = \langle R_{gp}\ R_{pm}\ L_m\ R_{gf}\ L_f\ L_g \rangle$, when $r = 1$ and $\ell = 2$.)

- Section 1 mentioned two ways of improving the expected cost of a derivation: (1) by determining the best strategy and (2) by adding redundancies. As empirical evidence has shown that using (2) without (1) can produce arbitrarily inefficient systems, this report has examined ways of combining both of these.

  Notice that one can, in theory, produce efficient systems by using (1) alone. These EBL-generated redundancies, however, can further improve the expected run-time costs of the overall system, if they are used properly.

- We have only been considering one type of EBL system, where the objective was finding the *optimal* strategy from a *redundant* knowledge base. An obvious variant of this task would involve determining whether a new redundant rule R_r, added to the front of the prior strategy, could *improve* the expected run-time of the overall system. (Of course, the new strategy would not include the rules with which R_r was redundant$_R$.) The obvious algorithm involves simply computing the expected costs of the two strategies and comparing the values. As computing costs can be done in linear time for our class of disjunctive knowledge bases, this overall system is linear.

  Another important class of EBL systems add *control information* rather than redundancies. These systems start with a given knowledge base K_1 and a specific initial strategy Θ_1; and then observe that the "derivation path" $\alpha = \langle R_1 \cdots L_n \rangle$ was successful in producing a solution to a given query. They form a new strategy Θ_2 by rearranging the rules and retrievals in Θ_1, moving α to front. (Notice no new rules are added to the system; *i.e.*, the knowledge base after the learning step is still K_1. The only change is to the derivation strategy.) Once again, there are two subcategories to consider: If the system insists that the new strategy, Θ_2, be optimal, then the task is obviously NP-hard whenever K_1 is redundant; see Theorem 3.1. Other systems may only consider whether $E[\Theta_2] < E[\Theta_1]$. Once again, one can obtain this answer in linear time for disjunctive knowledge bases— as it involves two linear-time cost computations.

4.2 Extensions

There are two obvious ways of extending this work. One involves coping with the *OptDS* task's inherent intractability by seeking efficient algorithms that return "approximately optimal" derivation strategies. [GO90] begins to address this task; see also [BJY89]. Another direction is to apply this same form of analysis to more general situations. The first obvious arena is handling conjunctive and recursive knowledge bases. Another is to combine this approach with other control strategy mechanisms — including conjunct ordering [SG85] and forward chaining [TG87]. The third is to obtain more accurate empirical values. For example, we have assumed that the costs of reductions and lookups (read "r" and "ℓ") are uniform for any rule or proposition. Preliminary empirical observations show that these costs depend on the number of variables, etc. A related point is our reliance on the "probability values" associated with each retrieval. [Smi89] uses the distribution of propositions in the knowledge base as an estimate for these values; another approach involves extrapolating these values from a set of observations, each of the form "query σ involved database fact τ" for some τ (see [GO90]).

4.3 Results

This research was motivated by the goal of evaluating the standard Explanation-Based Learning task, taking seriously the view that EBL is a method for improving the future performance of a reasoning system. This lead to the formal foundation for analysis presented in Section 2, based on the expected cost of solving a query from a given knowledge base, which uses the given probability that each proposition will be in the knowledge base. Section 3 uses this framework first to describe certain restricted situations where efficient algorithms are possible; and then to describe the computational complexities inherent in this undertaking. Section 4 uses this framework in an attempt to understand why EBL systems do, and do not, succeed in their attempts to improve the performance of their underlying systems. In particular, it proves the intractability of one type of EBL system, by observing that the general problem — which can involve conjunctions, recursion, etc. — must be at least as difficult as this simpler disjunctive case, which is already NP-hard.

Acknowledgements

Joe Likuski suggested many of the ideas presented here. David Smith provided extensive comments on earlier drafts of this paper, as did Cindy Chow, Hector Levesque, Bart Selman and Dale Schuurmans. Steven Rudich suggested the reduction to the Steiner Tree problem. This research was supported by an Operating Grant from the National Science and Engineering Research Council of Canada.

References

[BJY89] Danilo Bruschi, Deborah Joseph, and Paul Young. A structural overview of np optimization problems. In *Proceedings of the Second International Symposium on Optimal Algorithms*. Springer-Verlag Lecture Notes in Computer Science, 1989.

[Cho90] Cindy Chow. Obtaining an efficient derivational strategies for a conjunctive search space. Master's thesis, University of Toronto, 1990.

[DM86] Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–76, 1986.

[FHN72] Richard Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence: An International Journal*, 3:251–288, July 1972.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[GN87] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[GO90] Russell Greiner and Pekka Orponen. Probably approximately optimal satisficing strategy. Technical Report KRR-90-1, University of Toronto, 1990.

[Gre89] Russell Greiner. Finding the optimal derivation strategy in a redundant knowledge base. Technical Report KRR-TR-89-9, University of Toronto, August 1989.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[Lik88] Joseph Likuski. Integrating redundant learned rules in a knowledge base. Master's thesis, University of Toronto, October 1988.

[LNR86] John E. Laird, Allan Newell, and Paul S. Rosenbloom. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.

[Min85] Steven Minton. Selectively generalizing plans for problem solving. In *IJCAI-85*, pages 596–99, Los Angeles, August 1985.

[Min88] Steven Minton. Quantitative results concerning the utility of explanation-based learning. In *AAAI-88*, pages 564–69, San Mateo, CA, August 1988. Morgan Kaufmann Publishers, Inc.

[MKKC86] Thomas M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Example-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[RN82] Paul S. Rosenbloom and Allan Newell. Learning by chunking: Summary of a task and a model. In *AAAI-82*, Pittsburgh, August 1982.

[Ros83] Paul S. Rosenbloom. *The Chunking of Goal Hierarchies: A Model of Practice and Stimulus-Response Compatibility.* PhD thesis, Carnegie-Mellon University, August 1983.

[Rus85] Stuart Russell. *The Compleat Guide to MRS*, June 1985. Stanford KSL Report HPP-85-12.

[SG85] David E. Smith and Michael R. Genesereth. Ordering conjunctive queries. *Artificial Intelligence: An International Journal*, 26(2):171–215, May 1985.

[SK75] H. A. Simon and J. B. Kadane. Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence: An International Journal*, 6:235–247, 1975.

[Smi89] David E. Smith. Controlling backward inference. *Artificial Intelligence: An International Journal*, 39(2):145–208, June 1989. (Also Stanford Technical Report LOGIC-86-68).

[TG87] Richard J. Treitel and Michael R. Genesereth. Choosing orders for rules. *Journal of Automated Reasoning*, 3(4):395–432, December 1987.