Proofs for Symbolic Minimization

I. Symbolic Minimization of Relational DB is NP-Complete

Definition 1 (SMDB, decision version)**.** *Let $\mathcal{D}$ be a relational database and k be a positive integer. The minimization on $\mathcal{D}$ given k is to determine whether there is a triple $(\mathcal{H},\mathcal{N},\mathcal{C})$ such that:*

- $\|\mathcal{H}\| + |\mathcal{N}| + |\mathcal{C}| \leq k$
- $\mathcal{N} \subseteq \mathcal{D}$
- $\mathcal{N} \models^*_{\mathcal{H}} (\mathcal{D} \setminus \mathcal{N}) \cup \mathcal{C}$
- $\forall e \notin \mathcal{D} \cup \mathcal{C}, \mathcal{N} \not\models^*_{\mathcal{H}} e$

Theorem 2. *SMDB is in NP for fixed $\|\mathcal{H}\|$.*

Proof. According to the definition, $\mathcal{H}$ is a Datalog program. Let $\mathcal{A}$ be the evaluation result of $\mathcal{H}$ on $\mathcal{N}$. $(\mathcal{H},\mathcal{N},\mathcal{C})$ is a valid solution if the following are all true:

- $\|\mathcal{H}\| + |\mathcal{N}| + |\mathcal{C}| \leq k$
- $\mathcal{N} \subseteq \mathcal{D}$
- $\mathcal{A} \cap \mathcal{D} \supseteq \mathcal{D} \setminus \mathcal{N}$
- $\mathcal{A} \setminus \mathcal{D} = \mathcal{C}$

All above comparison can be finished in polynomial-time w.r.t. $\mathcal{D}, \mathcal{N}, \mathcal{C}, \mathcal{H}$ and $\mathcal{A}$. Moreover, $\mathcal{A}$ is computable in polynomial-time of $\mathcal{N}$ for fixed $\mathcal{H}$ [1]. Therefore, for fixed $\|\mathcal{H}\|$, the overall verification of the solution is in polynomial-time w.r.t. $\mathcal{D}$, $\mathcal{N}$ and $\mathcal{C}$. □

Definition 3 (Vertex Cover Problem)**.** *Let $\mathcal{G}_{vc} = \langle \mathcal{V}_{vc}, \mathcal{E}_{vc} \rangle$ be an undirected graph. Let k be a positive integer. A vertex cover $\mathcal{V}_c$ of $\mathcal{G}_{vc}$ is a subset of $\mathcal{V}_{vc}$ such that $(u,v) \in \mathcal{E}_{vc} \implies u \in \mathcal{V}_c \vee v \in \mathcal{V}_c$. The vertex cover problem is to determine whether there is a vertex cover $\mathcal{V}_c$ of $\mathcal{G}_{vc}$ s.t. $|\mathcal{V}_c| \leq k$.*

Hardness of the symbolic minimization can be proved by reducing the vertex cover problem to SMDB. Let $\mathcal{G}_{vc} = \langle \mathcal{V}_{vc}, \mathcal{E}_{vc} \rangle$ be the graph in the vertex cover problem and $n = |\mathcal{V}_{vc}|$, $m = |\mathcal{E}_{vc}|$. Let m_v be the number of edges connected to vertex v. By the following settings we create a relational database aligning with $\mathcal{G}_{vc}$:

- Let $edge$ be a unary relation in $\mathcal{D}$ for edges and v be a unary relation for each $v \in \mathcal{V}_{vc}$;
- For each $(u,v) \in \mathcal{E}_{vc}$, add three predicates to $\mathcal{D}$: $edge(e)$, $u(e)$, $v(e)$;
- Add $m+2$ predicates: $f_1(c_1)$, …, $f_{m+2}(c_{m+2})$.

The auxillary predicates $f_i(c_i)$ are used to make sure that Lemma 5 holds. If they are not included in the converted DB, other forms of rules, such as "$edge(X) \leftarrow true$", will also be feasible to reduce the size of the DB. The number of predicates, predicate symbols and constant symbols in $\mathcal{D}$ are $4m+2$, $m+n+3$ and $2m+2$. Therefore, the reduction can be done in linear time.

For example, Figure 1 shows a graph with three vertices and two edges. The corresponding database contains the following predicates:

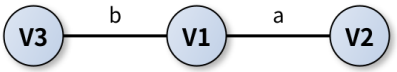

Fig. 1. Vertex Cover Example

- $edge(a)$, $edge(b)$
- $v_1(a)$, $v_2(a)$, $v_1(b)$, $v_3(b)$
- $f_1(c_1), \ldots, f_4(c_4)$

Lemma 4. *If $\mathcal{G}_{vc}$ can be covered by $\mathcal{V}_c$ s.t. $|\mathcal{V}_c| \leq k$, there is a solution $(\mathcal{H},\mathcal{N},\mathcal{C})$ for the corresponding minimization problem s.t. $\|\mathcal{H}\| + |\mathcal{N}| + |\mathcal{C}| \leq 3m+2+k$*

Proof. Let $\mathcal{H} = \{edge(X) \leftarrow v(X) | v \in \mathcal{V}_c\}$, $\mathcal{E} = \{edge(e) | e \in \mathcal{E}_{vc}\}$, $\mathcal{V} = \{v(e) | \exists u, e(u,v) \in \mathcal{E}_{vc}$ or $e(v,u) \in \mathcal{E}_{vc}\}$. $\mathcal{V} \models_{\mathcal{H}} \mathcal{E}$ and there is no counterexample as there is some $v \in \mathcal{V}_c$ that covers the corresponding edge. Moreover, $\mathcal{V} \subseteq \mathcal{D}$. Therefore, $(\mathcal{H}, \mathcal{D} \setminus \mathcal{E}, \varnothing)$ is a valid solution to SMDB. $\|\mathcal{H}\| + |\mathcal{D} \setminus \mathcal{E}| + |\varnothing| = |\mathcal{V}_c| \cdot 1 + 4m + 2 - m \leq 3m + 2 + k$. □

Let r be some inference rule. If for some $\mathcal{N} \subseteq \mathcal{D}$, $\mathcal{N} \models_r \mathcal{D} \setminus \mathcal{N}$, r separates $\mathcal{D}$ into two parts: $\mathcal{N}$ and $\mathcal{D} \setminus \mathcal{N}$, where the latter can be removed from $\mathcal{D}$. Let $\mathcal{C}$ be the set of all negatively inferred predicates from $\mathcal{N}$ w.r.t. r, $(\{r\},\mathcal{N},\mathcal{C})$ is a candidate solution (not necessarily minimum) to SMDB. The size reduced by r is:

$$\Delta(r) = |\mathcal{D}| - (|r| + |\mathcal{N}| + |\mathcal{C}|) = |\mathcal{D} \setminus \mathcal{N}| - |r| - |\mathcal{C}|$$

Lemma 5. *Considering all possible forms of a single inference rule, rules of the following form brings the most reduction on the database, which is $m_v - 1$:*

$$edge(X) \leftarrow v(X), m_v \geq 1 \tag{1}$$

Proof. Let r_v refer to the above rule. According to the construction of $\mathcal{D}$ and other definitions, the size reduced by a single rule r_v is $m_v - |r_v| = m_v - 1 \geq 0$ as there is no counterexample introduced by r_v. The reduction of the other forms of rules are:

- $\Delta(edge(?) \leftarrow true) = m - (m+2) = -2$;
- $\Delta(v(?) \leftarrow true) = m_v - (2m+2-m_v) = 2m_v - 2m - 2 \leq -2$;
- $\Delta(f_i(?) \leftarrow true) = 1 - (2m+1) = -2m \leq 0$;
- $\Delta(v(X) \leftarrow edge(X)) = m_v - 1 - m \leq -1$;
- $\Delta(v(X) \leftarrow u(X)) = b - 1 - (m_u - b) \leq 0$, where $b = [(u,v) \in \mathcal{E}_{vc} \vee (v,u) \in \mathcal{E}_{vc}]$, further more, when $b = 1$, $m_u \geq 1$, otherwise, $m_u \geq 0$;
- Given that only argument $v.0$ and $edge.0$ share some of constant symbols, any other longer forms of rules present no larger Δ than r_i above.

□

Lemma 6. *If $(\mathcal{H}, \mathcal{N}, \mathcal{C})$ is a solution to SMDB and $\|\mathcal{H}\| + |\mathcal{N}| + |\mathcal{C}| \leq k$, there is another solution $(\mathcal{H}', \mathcal{N}', \varnothing)$ to SMDB s.t.:*

- $|\mathcal{H}'| + |\mathcal{N}'| \leq k$
- *Rules in $\mathcal{H}'$ are all under the form of Rule (1);*
- *Let $\mathcal{E} = \{edge(e) | e \in \mathcal{E}_{vc}\}$, $\mathcal{N}' = \mathcal{D} \setminus \mathcal{E}$;*

Proof. Lemma 5 indicates that if some rule in $\mathcal{H}$ is not under the form of Rule (1), they can be simply removed without increasing the overall size, yielding $\mathcal{H}''$. All rules in $\mathcal{H}''$ are under the form of Rule (1). If $\exists edge(e_1), \ldots, edge(e_l) \in \mathcal{E}$ that are not inferable w.r.t. $\mathcal{H}''$, at most l rules under the form of Rule (1) that infer all of these predicates can be added to the hypothesis set, yielding $\mathcal{H}'$. The size of the minimization does not increase. After the above procedure, all edges are inferred by some vertex, thus $\mathcal{N}' = \mathcal{D} \setminus \mathcal{E}$. Moreover, by $\mathcal{H}'$, there is no counterexample in the solution. □

Lemma 7. *If $(\mathcal{H}, \mathcal{N}, \mathcal{C})$ is a solution to SMDB and $\|\mathcal{H}\| + |\mathcal{N}| + |\mathcal{C}| \leq 3m + 2 + k$, there is a $\mathcal{V}_c$ that covers $\mathcal{E}_{vc}$ and $|\mathcal{V}_c| \leq k$*

Proof. By Lemma 6, there is a solution $(\mathcal{H}', \mathcal{N}', \varnothing)$ with size no larger than $3m + 2 + k$. Let $\mathcal{V}_c = \{v | edge(X) \leftarrow v(X) \in \mathcal{H}'\}$. Vertices in $\mathcal{V}_c$ cover all edges as the rules in $\mathcal{H}'$ infer all edges in $\mathcal{D}$. $|\mathcal{V}_c| = |\mathcal{H}'| \leq 3m + 2 + k - |\mathcal{N}'| = k$. □

Theorem 8. *Symbolic minimization on relational database is NP-hard.*

Proof. By Lemma 4 and 7, the vertex cover problem has solution w.r.t. a positive integer k iff the reduced SMDB problem has solution w.r.t. $3m + 2 + k$. According to the reduction setting, the vertex cover problem can be polynomially reduced to SMDB. Therefore, SMDB is NP-Hard. □

Theorem 9. *Symbolic minimization on relational database is NP-Complete for fixed $\|\mathcal{H}\|$.*

Proof. According to Theorem 2 and 8, SMDB is in NP for fixed hypothesis and is also NP-Hard. Therefore, SMDB is NP-Complete for fixed $\|\mathcal{H}\|$. □

II. Properties of Horn Rules

A. Search Space for Rules

Definition 10 (Independent Fragments in Rules)**.** *An independent fragment in a rule r is a subset of the body that shares no variable with the remaining part of r (including the head of r).*

For example:

$$p(X) \leftarrow q(X, c, ?), s(Y, Z), t(Z, Y, ?) \tag{2}$$

$\{s(Y, Z), t(Z, Y, ?)\}$ is an independent fragment of Rule (2). This type of rules make nonsense since the variables in this subsequence are implicitly under existential quantifier, and:

- If the predicates in the fragment are satisfiable, then the rule is logically equivalent to the one without the independent fragment, which is not in the shortest form;
- If the predicates in the fragment are unsatisfiable, then the entire body of the rule will never be satisfied. And neither of positive nor negative entailment is inferable from the rule.

Neither of above cases is informative.

Let Ω be the search space for all first-order Horn rules. The new search space excluding rules with independent fragments is written as Ω_m.

B. Extension on Rules

Searching for rules starts from most general forms, i.e. rules only contain the head predicate and arguments in the predicate are all unique UVs. To construct new rules, new equivalence conditions are added to the equivalence classes. Syntactically, these operations fell in five extension operations, which are noted by $ext(r)$:

Case 1: Assign an existing LV to some argument.
Case 2: Add a new predicate with UVs to the rule and then assign an existing LV to one of these arguments.
Case 3: Assign a new LV to a pair of arguments.
Case 4: Add a new predicate with UVs to the rule and then assign a new LV to a pair of arguments. In this case, only one of the two arguments is selected from the newly added predicate.
Case 5: Assign a constant to some argument.

According to the rule extension, $\forall r, r_e \in \Omega_m$, if $r_e \in ext(r)$, r_e is the extension of r, and r is the origin of r_e (denoted as $r \in ext^{-1}(r_e)$ since one may have multiple origins). Neighbours of a rule in Ω_m consist of all its extensions and origins. The above extension operations can be used to search on Ω_m. Let $\mathcal{S} = \{p(\underbrace{?, \ldots, ?}_{\phi(p) \text{ times}}) \leftarrow | p \in \mathcal{P}\}$, every element in Ω_m can be searched from some $r_0 \in \mathcal{S}$. To prove this we define a property ***link*** between predicates in a certain rule:

Definition 11 (Link between Predicates)**.** *If two predicates p and q in a rule r share an LV X, p and q are linked by X in r, written as $p \diamond_X q$, or in short $p \diamond q$. Moreover, if there is a sequence of predicates $p \diamond w_0 \diamond \cdots \diamond w_k \diamond q$, then there is a linked path between p and q, written as: $p \leftrightarrow^{\diamond} q$.*

With this property, we can prove the search completeness as follows:

Lemma 12. *$\forall r \in \Omega_m$, every predicate in r has a linked path with the head of r.*

Proof. Suppose a predicate p in rule r has no linked path with the head. Then p is not itself the head. Let $\mathcal{Q} = \{q | p \leftrightarrow^{\diamond} q\}$, every predicate in $\mathcal{Q}$ has no linked path with the head. Then the fragment noted by $\mathcal{Q}$ does not share any variables with remaining predicates. Namely, $\mathcal{Q}$ denotes an independent fragment in rule r. According to the definition of Ω_m, we have $r \notin \Omega_m$, which contradicts with $r \in \Omega_m$. □

Lemma 13. *(Search Completeness) Let $\mathcal{S} = \{p(\underbrace{?, \ldots, ?}_{\phi(p) \text{ times}}) \leftarrow | p \in \mathcal{P}\}$, $\forall r \in \Omega_m, \exists r_0, r_1, \ldots, r_n \in \Omega_m$, such that $r_0 \in S, r_1 \in ext(r_0), \ldots, r \in ext(r_n)$.*

Proof. According to Lemma 12, all predicates in r has linked path with its head. Each predicate can be introduced into the

Algorithm 1 Minimization

Input: Database $\mathcal{D}$
Output: Minimization of $\mathcal{D}$: $(\mathcal{H}, \mathcal{N}, \mathcal{C})$

$\mathcal{H} \leftarrow \varnothing$
$\mathcal{C} \leftarrow \varnothing$
$\mathcal{G} \leftarrow \langle \mathcal{D} \cup \{\top\}, \varnothing \rangle$
while $true$ **do**
  $r \leftarrow findSingleRule(\mathcal{D})$
  if r cannot reduce the remaining of $\mathcal{D}$ **then**
    break
  end if
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{r\}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup E_r^-$
  Label $e \in E_r^+$ as inferred in $\mathcal{D}$
  Update graph $\mathcal{G}$ with respect to r
end while
$fvs \leftarrow FVS(\mathcal{G})$
$\mathcal{N} \leftarrow \{t \in V \setminus \{\top\} | \forall s \in V, (s,t) \notin E\} \cup fvs$
return $(\mathcal{H}, \mathcal{N}, \mathcal{C})$

rule following the linked path with head by case 2 and 4 extension operations. Other LVs and constants can be added to the rule by case 1, 3 and 5 extension operations to finally construct r. □

Rules with independent fragments will not be constructed starting from $\mathcal{S}$, as the extension operations do not introduce new predicates with no shared variables with other predicates.

Lemma 14. *The rule mining procedure terminates in finite steps.*

Proof. There is a finite number $k > 0$ such that for $|r| \geq k$, $\delta(r)$ is monotonically decreasing, because $|\mathcal{E}_r^+|$ and $|\mathcal{E}_r^-|$ both have lower bound 0 and finite upper bounds, and when the candidate rules are extended in Ω_m, both of $|\mathcal{E}_r^+|$ and $|\mathcal{E}_r^-|$ are non-increasing. Thus, for all rules with length no less than k, the maximum lies in the finitely many rules with length exactly k.

With in the part of search space such that $|r| < k$, there are only finitely many candidates for each given DB. Therefore, there is at least one local maximum in part of the search space where $|r| < k$. Overall, the rule mining procedure terminates in finite steps at some local maximum. □

III. Losslessness of Symbolic Minimization

Definition 15 (Inferable Predicate)**.** *Let $(\mathcal{H}, \mathcal{N}, \mathcal{C})$ be a solution to SMDB, $\mathcal{I}$ be a set of predicates, such that $\mathcal{I} = \mathcal{N} \cup \bigcup_{\mathcal{N} \models_{\mathcal{H}}^* \mathcal{T}} \mathcal{T}$. $\mathcal{I}$ is the set of predicates that can be recovered from the solution. A predicate P is said **inferable** w.r.t. $\mathcal{N}$ and $\mathcal{H}$ if and only if $P \in \mathcal{I}$.*

Definition 16 (Starting Set)**.** *Let $u \rightsquigarrow v$ denote a directed path from u to v in a directed acyclic graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. The **starting set** of v, written as $start(v)$, is the set of all starting endpoints of directed paths to v, i.e.,*

$$start(v) = \{u | u \rightsquigarrow v, \nexists u' \in \mathcal{V}, (u', u) \in \mathcal{E}\}$$

Lemma 17. *If $\mathcal{D} \subseteq \mathcal{I}$, every vertex in the dependency graph $\mathcal{G}_d$ with zero in-degree are included in $\mathcal{N}$.*

Proof. According to the definition of the dependency graph $\mathcal{G}_d$, at most one piece of evidence for each record is presented in $\mathcal{G}_d$ if the record can be entailed w.r.t. some rules. Vertices in $\mathcal{G}_d$ with zero in-degree refer to the records that cannot be entailed by any rule in $\mathcal{H}$, i.e. these vertices cannot be in the set $\bigcup_{\mathcal{N} \models_{\mathcal{H}}^* \mathcal{T}} \mathcal{T}$ for any $\mathcal{N}$. Therefore, if $\mathcal{D} \subseteq \mathcal{I}$, the above vertices can only be in $\mathcal{N}$. □

Lemma 18. *Let $\mathcal{N}_0$ be the set of all vertices with zero in-degree in $\mathcal{G}_d$. If $\mathcal{G}_d$ is a DAG and $\mathcal{N} = \mathcal{N}_0$, then $\mathcal{D} \subseteq \mathcal{I}$.*

Proof. If $\mathcal{G}_d$ is a DAG, $\forall v \in \mathcal{D}, start(v) \subseteq \mathcal{N}_0 = \mathcal{N}$. According to Definition 15, v can be entailed by $start(v)$ following the directed paths from $start(v)$ to v. □

Lemma 19. *If there are cycles in $\mathcal{G}_d$, let $\mathcal{N}_c$ be a set of vertices that covers the cycles in $\mathcal{G}_d$ (i.e. at least one vertex in each cycle is in $\mathcal{N}_c$). If $\mathcal{N} = \mathcal{N}_0 \cup \mathcal{N}_c$, then $\mathcal{D} \subseteq \mathcal{I}$.*

Proof. If $\mathcal{G}_d$ is a DAG, $\forall v \in \mathcal{D}, v \in \mathcal{I}$ (Lemma 18). If $\mathcal{G}_d$ is not a DAG, $\mathcal{G}_d$ can be made acyclic by removing all in-edges of vertices in $\mathcal{N}_c$, and this does not change the record that vertex in $\mathcal{N}_c$ are inferable. After the removal, $\mathcal{G}_d$ is acyclic, and all vertices with zero in-degree are in $\mathcal{N}$. Therefore, $\forall v \in \mathcal{D}, v \in \mathcal{I}$. □

Theorem 20 (Losslessness of Symbolic Minimization)**.** *Let $\mathcal{D}$ be a relational DB, $(\mathcal{H}, \mathcal{N}, \mathcal{C})$ be the minimization of $\mathcal{D}$ produced by Algorithm 1. $\mathcal{D} = \mathcal{I} \setminus \mathcal{C}$.*

Proof. According to Lemma 19, all records in $\mathcal{D}$ are inferable and can be recovered by iteratively evaluating $\mathcal{H}$ on $\mathcal{N}$ (the fixed-point semantics of $\models^*$). By the definition of $\mathcal{I}$, all negative entailments are included in $\mathcal{I}$, and so is $\mathcal{C}$. Therefore, $\mathcal{D} = \mathcal{I} \setminus \mathcal{C}$. □

IV. More Experimental Results

Besides the evaluations presented in the main text, we also tested our technique for its adaptability on different rule measures. The three assessment measures for rules are:

1) Minimization Capacity:

$$\delta(r) = |E_r^+| - |E_r^-| - |r|$$

2) Minimization Ratio:

$$\tau(r) = \frac{|E_r^+|}{|E_r^+| + |E_r^-| + |r|}$$

3) Information Gain:

$$h(r) = |E_r^+| \times \left(log \frac{|E_r^+|}{|E_r^+| + |E_r^-|} - log \frac{|E_{r'}^+|}{|E_{r'}^+| + |E_{r'}^-|} \right)$$

Table I displays summarization results under different assessment measures. All tests are run with the beam width set to 5. Induction under $\tau(r)$ tend to mine rules with higher accuracy-that is-with lower proportion of counter examples, or equivalently, more restrictions are included in the body.

TABLE I
SUMMARIZATION UNDER DIFFERENT MEASURES

Metric	δ				τ				h			
Dataset	θ (%)	#Assess.	Avg. $\|r\|$	Time (s)	θ (%)	#Assess.	Avg. $\|r\|$	Time (s)	θ (%)	#Assess.	Avg. $\|r\|$	Time (s)
Elti	66.42	22335	2.28	1.42	**41.33**	16583	1.89	1.12	56.83	7323	**1.32**	**0.80**
Dunur	81.36	43327	1.81	1.31	**75.91**	39792	1.81	1.61	88.18	18119	**1.29**	**0.95**
Student Loan	75.28	4440	1.73	**4.80**	**74.60**	7461	2.07	37.84	76.28	2296	**1.09**	5.99
DBpedia.recordbook	76.70	157	1.13	**0.20**	**58.35**	271	1.68	0.49	76.70	121	**0.92**	0.22
Family.simple	45.00	1736	1.45	**0.18**	45.00	1751	1.45	0.39	68.33	1168	**1.26**	0.27
Family.medium	52.91	20300	1.83	3.60	**40.09**	18694	1.78	3.83	88.00	5581	**1.30**	**2.60**
UMLS	39.06	945477	2.08	134	**32.16**	928297	2.10	137	95.34	92788	**1.01**	**6**

*Numbers in **black bold** is compared minimum; numbers with underline is compared maximum.

For example, to summarize relation "sameAs" in DBpedia.recordbook, $\tau(r)$ leads to Rule (3) other than (4), which is given by $\delta(r)$.

$$sameAs(Y, X) \leftarrow sameAs(X, Z), sameAs(Z, Y) \quad (3)$$

$$sameAs(X, X) \leftarrow \quad (4)$$

Summarization under $h(r)$ shows a much faster convergence. It tries to find rules that best distinguish positive and negative examples. In most cases, rules summarized under information gain are more general than $\tau(r)$ and $\delta(r)$ such that more counter examples are entailed. For example, information gain results in Rule (5) for "Family.medium", while $\delta(r)$ does Rule (6).

$$sister(X, Y) \leftarrow brother(Y, X) \quad (5)$$

$$sister(X, Y) \leftarrow aunt(X, Z), mother(Y, Z), parent(X, ?) \quad (6)$$

$\delta(r)$ balances between generality of rules and length of search paths. The average length of evaluated rules from $\delta(r)$ is shorter than $\tau(r)$ but longer than $h(r)$ as shown in Table I.

REFERENCES

[1] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, "Complexity and expressive power of logic programming," *ACM Computing Surveys (CSUR)*, vol. 33, no. 3, pp. 374–425, 2001.