VO Deductive Databases

WS 2014/2015

Stefan Woltran

Institut für Informationssysteme
Arbeitsbereich DBAI

Program Transformations

- **Basic Goal:** We are looking for *efficient* methods to replace in programs,
  - a set of rules R
  - by a *simpler* set of rules R',

  such that the answer sets are not changed by this manipulation.

- In particular,
  - "efficient" means that the method should be easier than checking equivalence between P and $(P \setminus R) \cup R'$, in general.
  - "simpler" refers to R' to be from an easier syntactical class than R, or to have less rules than R, ...

Program Transformations (ctd.)

- Formally, we consider triples $S : R \Rightarrow R'$ where,
  - R and R' are sets of rules;
  - S is the so-called *precondition*, i.e., a set of programs which have R as a subprogram.
- Let T be a triple of form $S : R \Rightarrow R'$.
  - A program P is called T-*applicable* iff $P \in S$;
  - For any T-applicable program P, $(P \setminus R) \cup R'$ is called the T-*result* of P.
  - T is called a *transformation*, iff, any T-applicable program P is equivalent to $(P \setminus R) \cup R'$.
  - If each program P with $R \subseteq P$ is contained in S, we leave S implicit, and identify T as the pair $R \Rightarrow R'$.

Program Transformations (ctd.)

- We call such transformations $R \Rightarrow R'$ (i.e., without precondition) also *local transformations*, since we replace R by R' without looking at the applied program, expect checking $R \subseteq P$.

- Transformations of the form $S : R \Rightarrow \emptyset$ are called *rule eliminations*; that is, R is deleted from an applied program P.

Program Transformations (ctd.)

- Observation: Local transformations inherently satisfy the condition that the result is *strongly* equivalent to the applied program.
- Formally, let $R \Rightarrow R'$ be a local transformation. Then, for any P with $R \subseteq P$, $P \equiv (P \setminus R) \cup R'$.
- Proof Sketch:
  - Any local transformation requires $R \equiv_s R'$, otherwise there exists at least one P, such that $AS(R \cup P) \neq AS(R' \cup P)$; but then applying $R \Rightarrow R'$ to the program $R \cup P$ would change the answer sets.
  - By definition of strong equivalence, $R \equiv_s R'$ implies that $P \equiv_s (P \setminus R) \cup R'$ holds, for any P with $R \subseteq P$.

Local Rule Elimination

- We consider local transformations of the form $R \Rightarrow \emptyset$.
- Observation: $R \Rightarrow \emptyset$ is a transformation iff, for each $r \in R$, $\{r\} \Rightarrow \emptyset$ is a transformation.
- It thus is sufficient to consider single rules which can be eliminated in any program, in order to get a full picture of local rule eliminations.
- In other words, we seek for rules which are strongly equivalent to the empty program.
  - ☞ Recall: the empty program has any SE-interpretation as its SE-model.

Local Rule Elimination (ctd.)

- **Proposition** [Osorio *et al.*, 01]: Any propositional rule r with

$$B^+(r) \cap \big(H(r) \cup B^-(r)\big) \neq \emptyset \tag{1}$$

  satisfies $\{r\} \equiv_s \emptyset$.

- Proof (Sketch). Each (J, I) with $J \subseteq I$ is SE-model of $\emptyset$. Towards a contradiction, suppose an SE-interpretation $(J, I) \notin SE(r)$ (if already $(I, I) \notin SE(r)$, use $J = I$ below). Then,

  (i) $I \cap B^-(r) = \emptyset$;

  (ii) $B^+(r) \subseteq J$, and

  (iii) $J \cap H(r) = \emptyset$.

  have to hold. Since r satisfies (1), either

  (a) $B^+(r) \cap H(r) \neq \emptyset$ *or* (b) $B^+(r) \cap B^-(r) \neq \emptyset$

  holds. But (a) is in contradiction to (ii)+(iii), and by $J \subseteq I$, (b) is in contradiction to (i)+(ii).

Local Rule Elimination (ctd.)

- It can be shown that the condition from the previous slide captures *all* possible local rule eliminations in the propositional setting.
- **Proposition** [Inoue and Sakama, 04]: Let r be a propositional rule. Then, $\{r\} \equiv_s \emptyset$ implies that $B^+(r) \cap \left(H(r) \cup B^-(r)\right) \neq \emptyset$ holds .
- We conclude: The set of all local rule eliminations in propositional ASP is exactly given by the set

$$\left\{R \Rightarrow \emptyset \mid \text{each } r \in R \text{ satisfies } B^+(r) \cap \left(H(r) \cup B^-(r)\right) \neq \emptyset\right\}.$$

Local Rule Elimination (ctd.)

- Interestingly, exactly the same condition applies to *non-ground* programs, i.e., programs with variables:
- **Proposition** [Eiter *et al.*, 06]: Let r be a non-ground rule. Then, $\{r\} \equiv_s \emptyset$ holds iff $B^+(r) \cap \big(H(r) \cup B^-(r)\big) \neq \emptyset$.
- In other words, the set of all local rule eliminations in ASP is exactly given by the set

$$\Big\{R \Rightarrow \emptyset \mid \text{each } r \in R \text{ satisfies } B^+(r) \cap \big(H(r) \cup B^-(r)\big) \neq \emptyset\Big\}.$$

Local Rule Elimination (ctd.)

- Examples:
  - We can remove rules of the form

    $$a(X) \vee b(Y,Z) \leftarrow c(X,Y), b(Y,Z)$$

    or

    $$a(X) \leftarrow b(X,Y), c(Z), not\ c(Z)$$

    from any program.
  - This does not holds for rules like,

    $$a(X) \vee b(Y,Z) \leftarrow c(X,Y), b(Z,Y)$$

    or

    $$a(X) \leftarrow b(X,Y), c(Z), not\ c(Y).$$

Local Rule Redundancy

- We now seek for translations of the form $\{r, s\} \Rightarrow \{s\}$.

- In other words, such translations allow us to eliminate a rule r, whenever an additional rule s is contained in the applied program.

- In our setting, such translations could also be represented using a precondition, i.e., using $S : \{r\} \rightarrow \emptyset$, with $P \in S$ iff $s \in P$ (and by definition, $r \in P$).

Local Rule Redundancy (ctd.)

- Example: Consider s to be the the rule $a \leftarrow$. Then, rules

$$a \vee b \leftarrow, \quad a \leftarrow b, \quad \text{or} \quad a \leftarrow not\ b$$

can faithfully be eliminated from any program containing s.

- This also holds for rules where a "moves" from the head to the negative body:

$$\leftarrow not\ a, \quad b \leftarrow not\ a, \quad \text{or} \quad \leftarrow b, not\ a.$$

- General Observation: For any rules r, s, the pair $\{r, s\} \Rightarrow \{s\}$ is a translation, iff $SE(s) \subseteq SE(r)$.

Local Rule Redundancy (ctd.)

- Rules r, s which satisfy $SE(s) \subseteq SE(r)$ have been characterized in [Lin and Chen, 05].

- **Proposition.** Let s and r be propositional rules, such that

$$H(s) \subseteq \big(H(r) \cup B^{-}(r)\big); \quad B(s) \subseteq B(r). \tag{2}$$

  Then, $SE(s) \subseteq SE(r)$.

- Further example for a translation $\{r, s\} \Rightarrow \{s\}$:

$$s = a \vee c \leftarrow b \quad \text{and} \quad r = a \leftarrow b, d, not\ c.$$

- Do rules r, s of form (2) characterize *all* possible transformations $\{r, s\} \Rightarrow \{s\}$? Not yet; we also need the case, where r can be eliminated anyway, i.e., where $\{r\} \Rightarrow \emptyset$ is already a transformation.

Local Rule Redundancy (ctd.)

- Let $\mathcal{R}$ be the set of all pairs $\{r, s\} \Rightarrow \{s\}$, such that either
  - $B^+(r) \cap \big(H(r) \cup B^-(r)\big) \neq \emptyset$, or
  - $H(s) \subseteq \big(H(r) \cup B^-(r)\big)$ and $B(s) \subseteq B(r)$ jointly hold.

- All local rule redundancies of the form $\{r, s\} \Rightarrow \{s\}$ are exactly given by $\mathcal{R}$.

- Note that given a program P, checking whether such a local redundancy can be applied to P, is computationally easy.

Local Rule Redundancy (ctd.)

- In the non-ground case, things become a bit more difficult.
  - First, we may have different variables in different rules:

    Example: $a(X) \leftarrow b(X)$ vs. $a(Y) \leftarrow b(Y), c(Z)$.

  - Second, a rule is also redundant if it has “less instantiations”:

    Example: $a(X) \leftarrow b(X)$ vs. $a(c) \leftarrow b(c)$.

Local Rule Redundancy (ctd.)

- Let $\mathcal{R}'$ be the set of all pairs $\{r, s\} \Rightarrow \{s\}$ of non-ground rules, such that there
  - exists a substitution $\theta : \mathcal{V}_s \to \mathcal{V}_r \cup \mathcal{C}_r$ satisfying
  - $H(s\theta) \subseteq \big(H(r) \cup B^-(r)\big)$ and $B(s\theta) \subseteq B(r)$ jointly hold.

- **Proposition** [Eiter *et al.*, 06]: Each element $\{r, s\} \Rightarrow \{s\}$ from $\mathcal{R}'$ is a translation.

- Given a program P, checking whether an element from $\mathcal{R}'$ can be applied to P is complete for NP.

Local Rule Redundancy (ctd.)

- Example: Consider

$$\begin{aligned} s &= p(X,Y) \leftarrow q(X), r(Y,Z) \\ r &= p(X,Y) \leftarrow q(X), r(Y,Y), r(Z,Z). \end{aligned}$$

r can be eliminated from any program containing s.

- Example: Consider

$$\begin{aligned} s &= a \leftarrow e(X_1,X_2), e(X_2,X_3), e(X_3,X_1) \\ r &= a \leftarrow e(g,r), e(r,g), e(r,b), e(b,r), e(g,b), e(b,g). \end{aligned}$$

r can be eliminated from any program containing s.

☞ Generalizing this example reduces 3-colorability (an NP-complete problem) to testing applicability of local rule redundancy.

Non-Local Transformations

- Next, we introduce a non-local transformation.
- Motivation: Consider the following two rules appear in a program:

$$\begin{array}{lcl} a & \leftarrow & b \\ a & \leftarrow & not\ b. \end{array}$$

  Can we simplify these two rules into a single rule, for instance $a \leftarrow$?
- Observation: $\{a \leftarrow\}$ is not strongly equivalent to $\{a \leftarrow b, a \leftarrow not\ b\}$:
  - $SE(\{a \leftarrow\}) = \{(a, a), (a, ab), (ab, ab)\}$; and
  - $SE(\{a \leftarrow b, a \leftarrow not\ b\}) = \{a, a), (a, ab), (ab, ab), (\emptyset, ab)\}$.

Non-Local Transformations (ctd.)

- However, $\{a \leftarrow\}$ and $\{a \leftarrow b, a \leftarrow not\ b\}$ are equivalent (they are also uniformly equivalent).
- This result extends to any program where b does not occur in rule heads, i.e., in each such program we can replace $\{a \leftarrow b, a \leftarrow not\ b\}$ by $\{a \leftarrow\}$ without changing the answer sets.
- In general, for any atom b, triples of the form $S_b : \{r, s\} \Rightarrow \{t\}$, where
  - r, s, t satisfy $b \in B^-(r) \cap B^+(s)$, $H(r) = H(s) = H(t)$, and $(B(r) \setminus \{not\ \ b\}) = (B(s) \setminus \{b\}) = B(t)$;
  - S_b is a set of programs with b not occurring in rule heads;

  are translations.

Non-Local Transformations (ctd.)

- A generalization to the non-ground case is as follows:
- For any non-ground atom b, triples $S_b : \{r, s\} \Rightarrow \{t\}$, where
  - for r, s, t, there exists a renaming $\theta : \mathcal{V}_r \rightarrow \mathcal{V}_s$, such that $b \in B^-(r\theta) \cap B^+(s)$, $H(r\theta) = H(s) = H(t)$, and $(B(r\theta) \setminus \{not\ \ b\}) = (B(s) \setminus \{b\}) = B(t)$;
  - each $P \in S_b$ satisfies: for each head atom a in P and each $\theta_a : \mathcal{V} \rightarrow \mathcal{C}$ and $\theta_b : \mathcal{V} \rightarrow \mathcal{C}$, $a\theta_a \neq b\theta_b$, i.e., a and b are not unifiable;

are translations.

Non-Local Transformations (ctd.)

- A further non-local transformation is *shifting* which eliminates disjunction under the precondition that the applied program is *head-cycle* free.

- Hereby, a proper disjunctive rule $a_1 \vee \cdots \vee a_n \leftarrow B(r)$, with $n > 1$, is replaced by a set of normal rules

$$\{a_i \leftarrow B(r), not\ a_1, \ldots, not\ a_{i-1}, not\ a_{i+1}, \ldots not\ a_n \mid 1 \leq i \leq n\}.$$

- Also shifting can be generalized to non-ground programs.
  - ☞ T. Eiter, M. Fink, H. Tompits, P. Traxler, and S. Woltran: Replacements in Non-Ground Answer-Set Programming. KR 2006.

Exercise

- We want to extend our results on *local rule redundancy* for the propositional case.

  Try to find patterns which yield (as many as possible) local transformations of the form $\{r, s, t\} \Rightarrow \{s, t\}$.

  Hint: Consider, e.g., $\{a \leftarrow c;\ a \leftarrow b;\ b \leftarrow c\} \Rightarrow \{a \leftarrow b;\ b \leftarrow c\}$.

Advertisement

- We are looking for people joining our team. Either if you are interested in
  - theoretical research; or
  - practical implementations;

  you are invited! We offer
  - support/supervision for your thesis (PhD, master, bachelor);
  - “Praktika” in this area;
  - participation in current research.