VO Deductive Databases

WS 2014/2015

Stefan Woltran

Institut für Informationssysteme
Arbeitsbereich DBAI

Comparing Propositional Programs

- Agenda:
  - Equivalence between Programs — Introduction;
  - Strong Equivalence;
  - Uniform Equivalence;
  - Further Notions of Equivalence.

Checking Equivalence—Motivation

- In ASP (generally, in any nonmonotonic formalism), it is to some extent unclear how to handle semantics of
  - program parts, as well as of
  - incomplete programs.

- This is because addition of further rules might withdraw previous conclusions.

- Instead of coming up with a concrete formal semantic treatment, one may consider the question, whether two different program fragments "*do the same job*" in a concrete scenario.

Checking Equivalence—Motivation (ctd.)

- Some important issues are closely related to this question:
  - simplification and (offline-)*optimization* issues;
  - *debugging* and *verification* features;
  - *modular logic programming*.

- Naive Approximation for "doing the same job":
  - (Ordinary) *equivalence* between two programs:

$$P \equiv Q \quad \text{iff} \quad P \text{ and } Q \text{ possess the same answer sets.}$$

Checking Equivalence—Motivation (ctd.)

- Due to nonmonotonicity of ASP, equivalence between programs is a much weaker concept than equivalence in classical logic.

- Consider $P = \{a \leftarrow b\}$ and $Q = \{a \leftarrow c\}$. We have

$$P \equiv Q \quad \text{but} \quad (P \cup \{b \leftarrow\}) \not\equiv (Q \cup \{b \leftarrow\}).$$

- Consider $P = \{a \leftarrow not\ b\}$ and $Q = \{a \leftarrow\}$. We have

$$P \equiv Q \quad \text{but} \quad (P \cup \{b \leftarrow\}) \not\equiv (Q \cup \{b \leftarrow\}).$$

- In general, equivalence in ASP does *not* satisfy the *replacement property*:

$$P \equiv Q \quad \text{implies} \quad R \equiv R[P/Q],$$

for any programs P, Q, and R.

Checking Equivalence—Motivation (ctd.)

- **Definition** [LPV 2001]: Two programs P, Q are *strongly equivalent* iff

$$(P \cup R) \equiv (Q \cup R) \quad \text{for any program } R.$$

  We write $P \equiv_s Q$ to denote that P and Q are strongly equivalent.

- Strong equivalence (SE) ensures the replacement property.

- It also has some nice computational properties.

- **But:** In many practical cases strong equivalence is much too restricted.

Checking Equivalence—Motivation (ctd.)

- Consider two different programs for (our running example) computing the accessible vertices:

$$
\begin{aligned}
P = \{\ & o(Y) \leftarrow v(X), e(X,Y);\\
& o(Y) \leftarrow o(X), e(X,Y)\} \text{ and}\\
Q = \{\ & p(X,Y) \leftarrow e(X,Y);\\
& p(X,Z) \leftarrow p(X,Y), e(Y,Z);\\
& o(Y) \leftarrow v(X), p(X,Y)\}.
\end{aligned}
$$

- Here we may want to compare P and Q wrt. a dedicated
  - *context*: add only programs over predicates $v(\cdot)$ and $e(\cdot,\cdot)$ to P, resp. Q;
  - *comparison relation* between the answer-sets (take only the "output" predicate $o(\cdot)$ into account).

Checking Equivalence—Motivation (ctd.)

- As a starting point, we shall focus on strong equivalence, however.
  - Important for efficient local optimization.
  - Provides a deeper understanding of ASP, in general.
  - Characterizations for strong equivalence are a basis for further notions of equivalence.

Strong Equivalence

- In what follows,
  - we focus (unless stated otherwise) on disjunctive programs;
  - $AS(P)$ denotes the set of all answer-sets of a program P.
- Recall: two programs P, Q, are strongly equivalent, $P \equiv_s Q$, iff $AS(P \cup R) = AS(Q \cup R)$ holds for any program R.
  - In order to decide strong equivalence, one wants to avoid testing equivalence for all program extensions R explicitly.
  - It turns out that an inspection of the models of the programs and their reducts is sufficient.

Strong Equivalence (ctd.)

- Basic observations: For any program P, and any interpretation I:
  - if $I \models P$, then I is answer set of $P \cup I$;
  - if $I \not\models P$, then I cannot be answer set for $P \cup R$, where R is any program.
- Examples:
  - Consider $\{p \leftarrow q\}$ and $I = \{p\}$. We have $I \models P$. I is answer set of $\{p \leftarrow q;\ p \leftarrow\}$, although I is not an answer set $\{p \leftarrow q\}$.
  - Consider the same program and $I = \{q\}$. Then, in any program containing rule $p \leftarrow q$, I cannot be an answer set.

Strong Equivalence (ctd.)

- Another result on program extension:
  Let P be a program, I an interpretation, and $J \subseteq I$. We have that
  - if $I \models P$ and $J \not\models P^I$, then I is a stable model of

$$R = P \cup J \cup \{p \leftarrow q \mid p, q \in (I \setminus J)\}.$$

  - Proof Sketch:
    We have $I \models R$. Also note that $R^I = P^I \cup J \cup \{p \leftarrow q \mid p, q \in (I \setminus J)\}$.
    For any $K \subset I$ we get, $K \not\models R^I$, since:
    1. if $J \not\subseteq K$: $K \not\models J$;
    2. if $K = J$: $K \not\models P^I$;
    3. otherwise, we have $J \subset K \subset I$: $K \not\models \{p \leftarrow q \mid p, q \in (I \setminus J)\}$ (note that $|I \setminus J| \geq 2$).

Strong Equivalence (ctd.)

- Definition:
  - An SE-interpretation (over $\mathcal{A}$) is a pair of interpretations (J, I), such that $J \subseteq I \subseteq \mathcal{A}$.
  - An SE-interpretation (J, I) is an SE-model of a program P iff $I \models P$ and $J \models P^I$.
  - The set of all SE-models of a program P is denoted by $SE(P)$.
- Definition: Call a program P *unary* iff each $r \in P$ is either a fact or of the form $p \leftarrow q$ (with p, q arbitrary atoms).
- Theorem: The following propositions are equivalent:
  1. $P \equiv_s Q$; i.e., for each program R, $AS(P \cup R) = AS(Q \cup R)$;
  2. for each unary program R, $AS(P \cup R) = AS(Q \cup R)$;
  3. $SE(P) = SE(Q)$.

  (Proof on blackboard).

Strong Equivalence (ctd.)

- **Example**: Let

$$P = \{a \lor b \leftarrow\} \quad \text{and} \quad Q = \{a \leftarrow not\ b;\ b \leftarrow not\ a\}.$$

- For P, we have the following classical models (over $\{a, b\}$):

$$\{a\},\ \{b\},\ \{a, b\}$$

- Thus candidates of the SE-models of P are $(\cdot, a)$, $(\cdot, b)$, and $(\cdot, ab)$.
  - ☞ With some abuse of notation, we skip "{" and "}" within SE-models.

- For any interpretation I, we have $P^I = P$ (since P is positive here). Hence,

$$SE(P) = \{(a, a), (b, b), (a, ab), (b, ab), (ab, ab)\}.$$

Strong Equivalence (ctd.)

- For $Q = \{a \leftarrow not\ b;\ b \leftarrow not\ a\}$, we have the same classical models (over $\{a, b\}$), namely $\{a\}$, $\{b\}$, $\{a, b\}$.
- But, now we have to take the respective reducts into account:
  - for $I = \{a\}$: models $J \subseteq I$ of $Q^I = \{a \leftarrow\}$ are $\{a\}$;
  - for $I = \{b\}$: models $J \subseteq I$ of $Q^I = \{b \leftarrow\}$ are $\{b\}$;
  - for $I = \{a, b\}$: models $J \subseteq I$ of $Q^I = \emptyset$ are $\emptyset, \{a\}, \{b\}, \{ab\}$.
- We get

$$SE(Q) \quad = \quad \{(a, a), (b, b), (\emptyset, ab), (a, ab), (b, ab), (ab, ab)\}.$$

- We have $(\emptyset, ab) \notin SE(P)$, but $(\emptyset, ab) \in SE(Q)$. Hence, $P \not\equiv_s Q$, as witnessed by the counter-example $(\{a, b\},\ \{a \leftarrow b, b \leftarrow a\})$:
  - $\{a, b\} \in AS(P \cup \{a \leftarrow b, b \leftarrow a\})$;
  - $\{a, b\} \notin AS(Q \cup \{a \leftarrow b, b \leftarrow a\}) = \emptyset$.

Strong Equivalence (ctd.)

- General definition: A *counter-example* (Y, R) to a SE-Test $P \equiv_s Q$ is given by
  - an interpretation Y, and a
  - a program R,

  such that, either
  - $Y \in AS(P \cup R)$ and $Y \notin AS(Q \cup R)$; or
  - $Y \in AS(Q \cup R)$ and $Y \notin AS(P \cup R)$.

Strong Equivalence (ctd.)

- Consider programs $P = \{a \leftarrow\}$ and Q $= \{a; a \leftarrow b; a \leftarrow \mathit{not}\ c\}$. The SE-models (over $\{a, b, c\}$) of both programs coincide and are given by

$$(a, a); (a, ab); (a, ac); (ab, ab); (ac, ac); (a, abc); (ab, abc); (ac, abc); (abc, abc).$$

- General observation: $\{r\}$ is strongly equivalent to any $\{r, s\}$, whenever $SE(r) \subseteq SE(s)$.
  - in particular, this holds for any rules r, s, such that

$$H(r) \subseteq H(s); \quad B(r) \subseteq B(s).$$

  - Such results provide the basis for (local) program simplification techniques: In any program, with rules r, s as above, one can faithfully delete rule s.

Strong Equivalence (ctd.)

- In general, we can decide SE via propositional logic:
- Recall (from 2nd lecture): For a program P be a program over atoms V, $J, K \subseteq V$; and I any interpretation, such that $(I \cap V) = J$ and $(I \cap V') = K'$, it holds that

  $$I \text{ is a model of } P^* \text{ iff } K \models P^J$$

  (where P^* was like $\{B^+(r') \wedge \neg B^-(r) \supset H(r') \mid r \in P\}$).

- **Proposition.** Let P, Q be be programs over atoms V, then $P \equiv_s Q$ iff the formula

  $$(V' \leq V) \supset \Big((\hat{P} \wedge P^*) \equiv (\hat{Q} \wedge Q^*)\Big)$$

  is valid.

Strong Equivalence (ctd.)

- Let P be a *normal* program. Then, its SE-models satisfy the following property (reduct-intersection):

$$(J, I) \in SE(P) \text{ and } (K, I) \in SE(P) \text{ then } (J \cap K, I) \in SE(P)$$

  - Since for any I, P^I is a Horn program.

- If the SE-models of a disjunctive program do not satisfy reduct-intersection, then no strongly equivalent normal programs exists.
  - Hence, given a disjunctive program P, reduct-intersection on $SE(P)$ provides a *necessary* condition, for the question whether there exists a normal program Q, such that $P \equiv_s Q$. It can be shown that this condition is also *sufficient* for this problem.

Strong Equivalence (ctd.)

- Recall example $\{a \vee b \leftarrow\}$. We have as its SE-models

$$(a, a), (b, b), (a, ab), (b, ab), (ab, ab).$$

They do not satisfy reduct-intersection, since (a, ab) and (b, ab) call for $(\emptyset, ab)$.

➥ No normal program is strongly equivalent to $\{a \vee b \leftarrow\}$.

Strong Equivalence (ctd.)

- Some complexity results:
  - Checking strong equivalence between disjunctive programs P, Q is coNP-complete; hardness holds already for the case that P is normal and Q is Horn.
    1. Membership follows immediately from our reduction to propositional validity.
    2. Hardness (of the complementary problem) uses the same construction as in the proof for NP-hardness of deciding whether a normal program has at least a stable model; compare this program to the (Horn) program: $\{\bot \leftarrow\}$.
  - Checking reduct-intersection is coNP-complete.

Uniform Equivalence

- Considering programs as database queries the following notions are more natural:
- Two programs P, Q are *uniformly equivalent* iff,

$$AS(P \cup F) = AS(Q \cup F) \quad \text{for any set } F \text{ of facts.}$$

- Traditional database view: Call atoms which occur only in rule-bodies of a program *external*. Two programs P, Q are *program equivalent* (or *query equivalent*) iff,

$$AS(P \cup E) = AS(Q \cup E) \quad \text{for any set } E \text{ of external atoms.}$$

- We have the following implications:
  - strong equivalence implies uniform equivalence;
  - uniform equivalence implies program equivalence.

Uniform Equivalence (ctd.)

- Given a program P. An SE-interpretation (J, I) is an UE-model of P iff
  - (J, I) is an SE-model of P; and,
  - for each K with $J \subset K \subset I$, (K, I) is not SE-model of P.
- Hence UE-models of a program P, denoted $UE(P)$, are
  - all total SE-models (I, I) of P,
  - all further SE-models (J, I) of P, where $J \subset I$ is maximal in being model of P^I.
- **Proposition**. Two programs P, Q are uniformly equivalent iff $UE(P) = UE(Q)$.

Uniform Equivalence (ctd.)

- Example: Let

$$P = \{a \vee b \leftarrow\} \text{ and } Q = \{a \leftarrow not\ b;\ b \leftarrow not\ a\}.$$

  We have

  - $UE(P) = SE(P) = \{(a, a), (b, b), (a, ab), (b, ab), (ab, ab)\}$;
  - $UE(Q) = (SE(Q) \setminus \{(\emptyset, ab)\}) = \{(a, a), (b, b), (a, ab), (b, ab), (ab, ab)\}$.

- Hence, P and Q are uniformly equivalent, although they are not strongly equivalent.

Uniform Equivalence (ctd.)

- Complexity of checking uniform equivalence (UE).
  - UE between disjunctive programs is Π_2^P-complete;
  - UE between normal programs is coNP-complete.
- Source of complexity (for disjunctive programs):
  - given (J, I), checking whether (J, I) is UE-model of P is already coNP-complete (due to test for maximality).

Equivalence with Projection

- It is often desired to compare the outcome of programs only on a subset of the atoms involved (output-predicates).

- For instance, let P, Q be programs over $\mathcal{A}$ and $O \subseteq \mathcal{A}$. Then,

$$P \equiv_O Q \quad \text{iff} \quad \{(I \cap O) \mid I \in AS(P)\} = \{(J \cap O) \mid J \in AS(Q)\}.$$

- Example: Consider programs for guessing any subset of $\{p, q\}$.

$$P = \{ \begin{array}{l} p \vee p' \leftarrow \\ q \vee q' \leftarrow \} \end{array} \qquad Q = \{ \begin{array}{l} p \vee \bar{p} \leftarrow \\ q \vee \bar{q} \leftarrow \} \end{array}$$

and $O = \{p, q\}$. Then $P \equiv_O Q$ holds.

Equivalence with Projection (ctd.)

- In general, projection is an additional source of complexity.
- **Theorem.** Given disjunctive programs P, Q, and a set of atoms O, deciding $P \equiv_O Q$ is Π_3^P-hard.
  - Membership: We show the complementary problem to be in Σ_3^P.
    Let P and Q be given over atoms $\mathcal{A}$ and $O \subseteq \mathcal{A}$.
    Guess I and check whether
    1. $I \in \mathcal{AS}(P)$;
    2. the program

       $$Q \cup \{\bot \leftarrow v \mid v \in (O \setminus I)\} \cup \{\bot \leftarrow not\ u \mid u \in (I \cap O)\}$$

       has no stable model;

    or vice versa (Note: 1. is in coNP; 2. is in Π_2^P; together with the guess, we obtain membership in Σ_3^P).
  - Hardness (blackboard!)

Equivalence with Projection (ctd.)

- Interestingly, in the case of strong equivalence, using projection does not result in an increase of complexity.
- Define $P \equiv_{s,O} Q$ iff for each program R, $(P \cup R) \equiv_O (Q \cup R)$.
- **Proposition.** Let P and Q be programs over $\mathcal{A}$. For any set of atoms $O \subseteq \mathcal{A}$, it holds that $P \equiv_{s,O} Q$ iff $P \equiv_s Q$.
  - only-if: Suppose (I, R) is a counter-example of $P \equiv_s Q$. Then

    $$\big(I,\;\; R \cup \{\bot \leftarrow v \mid v \in (\mathcal{A} \setminus I)\} \cup \{\bot \leftarrow \mathit{not}\ u \mid u \in I\}\big)$$

    is a counter-example of $P \equiv_{s,O} Q$.
  - if: by definition.

➥ Deciding $P \equiv_{s,O} Q$ is coNP-complete.

Further Notions

- In the literature further notions are proposed and investigated:
  - Strong equivalence relative to a set of atoms A: Given P, Q, does $(P \cup R) \equiv (Q \cup R)$ hold for all programs R over A?
  - Uniform equivalence relative to a set of atoms A: Given P, Q, does $(P \cup R) \equiv (Q \cup R)$ hold for all sets R of facts from A?
    - ➥ Setting A as the set of external atoms in $P \cup Q$ yields program equivalence.
  - Combination: A, O-equivalence, for sets of atoms A, O: Given P, Q, does $(P \cup R) \equiv_O (Q \cup R)$ hold for all programs R over A?
    - ☞ This combination yields Π_4^P-hardness.

Further Notions (ctd.)

- Head-Body Relativized Equivalence [TPLP, 2008]:
  - Idea: Equivalence notion is specified by two parameters; one alphabet for atoms in heads and one for atoms in bodies.
  - Generalizes other notions of equivalence introduced so far.
  - Let $A, B \subseteq \mathcal{A}$. Then $\mathcal{C}(A, B)$ is the set of all programs P such that $H(P) \subseteq A$ and $B(P) \subseteq B$ (atoms in heads are from A; atoms in bodies are form B).
  - Programs P and Q are (A, B)-equivalent, if for every program $R \in \mathcal{C}(A, B)$, $AS(P \cup R) = AS(Q \cup R)$.

Literature

- Complexity:

  T. Eiter, G. Gottlob: *On the Computational Cost of Disjunctive Logic Programming: Propositional Case*. Ann. Math. Artif. Intell. 15(3-4): 289–323 (1995).

  T. Eiter, M. Fink, S. Woltran: *Semantical Characterizations and Complexity of Equivalences in Answer Set Programming*, ACM Trans. Comp. Logic 8(3), 2007.

- Logical Underpinnings of Strong Equivalence:

  V. Lifschitz, D. Pearce, A. Valverde: *Strongly Equivalent Logic Programs*. ACM Trans. Comput. Log. 2(4): 526–541 (2001).

  D. De Jongh, L. Hendriks: *Characterization of Strongly Equivalent Logic Programs in Intermediate Logics*. TPLP 3(3): 259–270 (2003).

- Equivalence Notions with Projection:

  T. Eiter, H. Tompits, S. Woltran: *On Solution Correspondences in Answer Set Programming*. Proceedings IJCAI'05, 97–102, 2005.

  J. Oetsch, H. Tompits, S. Woltran: *Facts Do Not Cease to Exist Because They Are Ignored*. Proceedings AAAI'07, 458–464

Exercises

- Consider the programs $P = \{a \leftarrow b;\ a \leftarrow not\ b\}$ and $Q = \{a \leftarrow c;\ a \leftarrow not\ c\}$. Check whether P and Q are strongly equivalent or uniformly equivalent; and whenever this is not the case, provide a counter-example.

- Given a program P, provide a necessary condition for $SE(P)$ which has to hold, such that there exists a positive program strongly equivalent to P. Recall that for each positive program Q, $Q^I = Q$ holds for any interpretation I.