VO Deductive Databases

WS 2014/2015

Stefan Woltran

Institut für Informationssysteme
Arbeitsbereich DBAI

Propositional Answer-Set Programming

- Agenda:
  - Horn Programs;
  - Adding Negation;
  - Disjunctive Programs;
  - Further Classes and Extensions;
  - Relation between Answer-Set Programming and Classical Logic.

Definite Horn Programs—Introduction

- Recall from last lecture: Given a graph by its set of edges e, and a set of designated vertices d; the program

$$\begin{array}{rcl} out(Y) & \leftarrow & v(X), e(X,Y). \\ out(Y) & \leftarrow & out(X), e(X,Y). \end{array}$$

computes via $out(\cdot)$ all nodes reachable from the designated vertices.

- We may consider *ground* variants of such an application as follows:
  - Let $v_1, \ldots, v_n$ be potential nodes of graphs. Consider program

$$P_e \cup P_d \cup P_q$$

where

$$\begin{array}{rcl} P_e & \subseteq & \{e_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq n\}; \\ P_d & \subseteq & \{v_i \mid 1 \leq i \leq n\}; \\ P_q & = & \{o_j \leftarrow v_i, e_{i,j}; \\ & & \quad o_j \leftarrow o_i, e_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq n\}. \end{array}$$

Definite Horn Programs—Introduction (ctd.)

- Example: Graphs over two nodes v_1, v_2.
- Let us consider the simple graph with nodes v_1, v_2 having a directed edge from v_1 to v_2; with v_1 being designated.

  We get the following program:

$$
\begin{aligned}
\{ \quad & e_{1,2}; \\
& v_1; \\
& o_1 \leftarrow v_1, e_{1,1}; \\
& o_1 \leftarrow v_2, e_{2,1}; \\
& o_2 \leftarrow v_1, e_{1,2}; \\
& o_2 \leftarrow v_2, e_{2,2} \quad \}.
\end{aligned}
$$

  Intuitively, we would consider $\{v_1; e_{1,2}; o_2\}$ as intended model.

Definite Horn Programs—Introduction (ctd.)

- Central Observation. The intended model $\{v_1; e_{1,2}; o_2\}$ is given by the *minimal* classical model of the propositional theory

$$\begin{aligned}\{\quad & v_1;\ e_{1,2}; \\ & v_1 \wedge e_{1,1} \supset o_1; \\ & v_2 \wedge e_{2,1} \supset o_1; \\ & v_1 \wedge e_{1,2} \supset o_2; \\ & v_2 \wedge e_{2,2} \supset o_2\ \}.\end{aligned}$$

- Indeed, this theory has further (non-minimal) models, which are not intended.

Definite Horn Programs—Syntax

- A definite Horn rule r (over $\mathcal{A}$) is an expression of the form

$$h \leftarrow b_1, b_2, \ldots, b_n$$

where $h, b_1, \ldots, b_n$ are propositional atoms (from $\mathcal{A}$), and $n \geq 0$.

- Instead of "$h \leftarrow$" we sometimes simply write "h"; rules of this form are called *facts*.

- We call
  - $H(r) = \{h\}$ the *head* of r;
  - $B(r) = \{b_1, b_2, \ldots, b_n\}$ the *body* of r.

- A *definite Horn program* is a set of definite Horn rules.

Definite Horn Programs—Semantics

- Let r be a rule
$$h \leftarrow b_1, \ldots, b_n$$
over $\mathcal{A}$. An interpretation $I \subseteq \mathcal{A}$ is a *model* of r iff the following holds:
If $b_1, \ldots, b_n$ is in I, then $h \in I$.

- Define for r as above:
$$\hat{r} = b_1 \wedge \cdots \wedge b_n \supset h.$$
  - Then I is a model of a rule r iff I is a model of the formula $\hat{r}$.

- An interpretation $I \subseteq \mathcal{A}$ is a *model* of a definite Horn program P iff I is a model of each $r \in P$.
  - I is a model of a program P iff I is a model of the *associated theory* $\hat{P} = \{\hat{r} \mid r \in P\}$.

- We use $I \models r$ (resp. $I \models P$) to denote that I is model of r (resp. P).

Definite Horn Programs—Semantics (ctd.)

- For each definite Horn program P there is a unique minimal model.
- This follows from the fact that the models of P satisfy the *intersection* property (proof on blackboard):

  If I and J are models of P, then $(I \cap J)$ is a model of P.

- We call this minimal model of P, the *stable model* or the *answer set* of P.

Definite Horn Programs—Semantics (ctd.)

- Example: Consider the program $P = \{a;\ a \leftarrow b;\ a \leftarrow c\}$.
- P has models (over $\{a, b, c\}$): $\{a\}$, $\{a, b\}$, $\{a, c\}$, $\{a, b, c\}$.
- They satisfy the intersection property since, e.g.,

$$\{a, b\} \cap \{a, c\} = \{a\}.$$

Definite Horn Programs—Complexity

- **Proposition.** The minimal model of a definite Horn program can be computed in polynomial time.

- **Proposition.** The problem of deciding whether a given atom a is contained in the minimal model of a definite Horn program is P-complete.
  - membership is a direct consequence from first Proposition.
  - hardness: via an encoding of a DTM; rules represent transitions between states; ask whether an accepting state is reached.
    - ☞ this actually shows more than P-completeness; gives results in terms of *expressibility*, i.e., with respect to *search* problems.

Adding Negation—Introduction

- Recall our example on graphs. Consider we want to compute all vertices which are *not accessible* via designated vertices.
- Desired solution: Let us add negation, such that we can add rules of the form

$$\{u_i \leftarrow not\ o_i\}$$

  stating if vertex v_i is not accessible ($not\ o_i$), then u_i explictly marks that vertex as unaccessible.
- In our concrete example with vertices v_1, v_2, an edge from v_1 to v_2, and the designated node v_1, we would then consider as intended model: $\{v_1; e_{1,2}; o_2; u_1\}$.

Adding Negation—Introduction (ctd.)

- Further example: Compute all nodes which would be accessible in the "complement" $\overline{G}$ of a given graph G. ($\overline{G}$ has the same vertices V, but (v_i, v_j) is an edge in $\overline{G}$ iff (v_i, v_j) is not an edge in G).
- Solution: Replace the part P_q in the general encoding by

$$\{o_j \leftarrow v_i, not\ e_{i,j};\ \ o_j \leftarrow o_i, not\ e_{i,j} \mid 1 \leq i, j \leq n\}.$$

- Problem: What is the semantics of

$$\{\text{man};\ \ \text{single} \leftarrow \text{man}, not\ \text{husband};\ \ \text{husband} \leftarrow \text{man}, not\ \text{single}\}\ ?$$

Intended models: $\{\text{man}; \text{single}\}$ and $\{\text{man}; \text{husband}\}$.

Adding Negation—Introduction (ctd.)

- Let us consider the minimal models of the theory associated to the (simplified) program:

  $$\{s \leftarrow \mathit{not}\ h;\ h \leftarrow \mathit{not}\ s\} \quad \text{i.e.,} \quad \{\neg h \supset s,\ \neg s \supset h\}.$$

  The theory has three models $\{s\}$, $\{h\}$, and $\{s, h\}$ with the first two being minimal.

  - Ok.

- But: For the program $\{s \leftarrow \mathit{not}\ h\}$ we get the same models and thus the same minimal models as above.

  - Unintuitive!

Adding Negation—Introduction (ctd.)

- Great logic programming schism:
  1. Single intended model approach: Select a single model of all classical models.
  2. Multiple preferred model approach: Select a subset of all classical models.
- With a syntactic restriction (*stratification*—will be introduced later), we can use negation and retain the "single-model property".

Normal Programs—Syntax

- A normal rule r (over $\mathcal{A}$) is an expression of the form

$$h \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m$$

where $h, b_1, \ldots, b_m$ are propositional atoms (from $\mathcal{A}$), and $m \geq 0$.

- We call
  - $H(r) = \{h\}$ the *head* of r;
  - $B(r) = \{b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m\}$ the *body* of r.
  - $B^+(r) = \{b_1, \ldots, b_n\}$ the *positive body* of r;
  - $B^-(r) = \{b_{n+1}, \ldots, b_m\}$ the *negative body* of r.
- A *normal program* is a set of normal rules.

Normal Programs—Semantics

- Let r be a rule

$$h \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m$$

An interpretation $I \subseteq \mathcal{A}$ is a *model* of r iff the following holds:

If $b_1, \ldots, b_n$ are all in I, and none of $b_{n+1}, \ldots, b_m$ are in I then $h \in I$.

- Define for r as above:

$$\hat{r} = b_1 \wedge \cdots \wedge b_n \wedge \neg b_{n+1} \wedge \cdots \wedge \neg b_m \supset h.$$

  - Then I is a model of a rule r iff I is a model of the formula $\hat{r}$.

- An interpretation $I \subseteq \mathcal{A}$ is a *model* of a normal program P iff I is a model of each $r \in P$.

  - As before: I is a model of a program P iff I is a model of the *associated theory* $\hat{P} = \{\hat{r} \mid r \in P\}$.

- Again, $I \models r$ (resp. $I \models P$) denotes that I is model of r (resp. P).

Normal Programs—Semantics (ctd.)

- So far, we did not solve the problem involving negation!
- Solution (Gelfond and Lifschitz, 1988; Bidoit and Froidevaux, 1988):
  - Define a *reduct* of a program P with respect to some interpretation I:

$$P^I \;=\; \{H(r) \leftarrow B^+(r) \mid r \in P; (I \cap B^-(r)) = \emptyset\}$$

- Intuition:
  - I makes an *assumption* about what is true and what is false;
  - P^I derives positive information under the assumption of I, wrt to negative bodies;
  - if the "result" then is I itself, the assumption I is stable.

Normal Programs—Semantics (ctd.)

- Let I be an interpretation; P a normal program. Then, I is a *stable model* (or an *answer set*) of P iff I is a minimal model of P^I.
- Now, programs may have none, one, or more stable models!
- Example: $P = \{s \leftarrow not\ h\}$. We expect $\{s\}$ to be the only stable model. We check:
  - $I = \emptyset$; then $P^I = \{s\}$, but $I \not\models P^I$.
  - $J = \{s\}$; then $P^J = \{s\}$, $J \models P^J$ and is minimal! J **is stable.**
  - $K = \{h\}$; then $P^K = \emptyset$, but $\emptyset \subset K$ is model of P^K.

    Note: By definition, the empty program has any interpretation as its model.
  - $L = \{s, h\}$; then $P^L = \emptyset$, but $\emptyset \subset L$ is model of P^L.

Normal Programs—Semantics (ctd.)

- Example: $P = \{s \leftarrow \mathit{not}\ h;\ h \leftarrow \mathit{not}\ s\}$. We expect $\{s\}$ and $\{h\}$ to be the stable models of P. We check:
  - $I = \emptyset$; then $P^I = \{s;\ h\}$, but $I \not\models P^I$.
  - $J = \{s\}$; then $P^J = \{s\}$, $J \models P^J$ and is minimal! J **is stable.**
  - $K = \{h\}$; then $P^K = \{h\}$, $K \models P^K$ and is minimal! K **is stable.**
  - $L = \{s, h\}$; then $P^L = \emptyset$, but $\emptyset \subset L$ is model of P^L.
- Example: The program $\{p \leftarrow \mathit{not}\ p\}$ has *no* stable model.
  - $I = \emptyset$; then $P^I = \{p\}$, but $I \not\models P^I$.
  - $J = \{p\}$; then $P^J = \emptyset$ but $\emptyset \subset J$ is model of P^J.
    - ☞ Note that the associated theory has a classical model!

Normal Programs—Semantics (ctd.)

- Some observations:
  - A normal program without negation is a definite Horn program, and thus has a unique stable model.
  - For any interpretation I and any normal program P, P^I is a definite Horn program.
  - There may be an exponential number of stable models of a program compared to its size:

$$P = \{v_i \leftarrow not\ u_i;\ u_i \leftarrow not\ v_i \mid 1 \leq i \leq n\}$$

    has 2^n stable models.

Constraints

- Let P a program, q an atom not occurring in P and consider a rule

$$q \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m, not\ q.$$

  This rule "kills" all stable models of P, that

  - contain $b_1, \ldots, b_n$; and
  - do not contain $b_{n+1}, \ldots, b_m$.

- We abbreviate such rules by

$$\perp \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m$$

  and call them *constraints*.

The-Generate-and-Check Paradigm

- The first part of a program generates potential solution candidates.
- The second part rules out all candidates violating some condition to be a solution.
- Example: Graph 2-coloring. Given graph, can we assign to each vertex one color (say, either red or green) such that connected vertices do not have the same color:
  - Let a set of facts $e_{i,j}$ specify our graph over vertices $v_1, \ldots, v_n$.
  - Generate candidates:

$$\{r_i \leftarrow \mathit{not}\ g_i;\ g_i \leftarrow \mathit{not}\ r_i \mid 1 \leq i \leq n\}.$$

  - Check candidates:

$$\begin{aligned} \{ \quad & \bot \leftarrow e_{i,j}, r_i, r_j; \\ & \bot \leftarrow e_{i,j}, g_i, g_j \mid 1 \leq i \leq n; 1 \leq j \leq n\}. \end{aligned}$$

Horn Programs

- A Horn program is a definite Horn program plus a set of positive constraints (i.e., without negative body-atoms).

- Checking whether a Horn program P has a stable model is decidable in polynomial time:
  - Compute the unique minimal of the definite Horn part.
  - Check whether this model passes through the constraints.

Normal Programs—Complexity

- Checking whether a normal program P has at least one stable model is NP-complete
  - Membership.
    1. Guess an interpretation I;
    2. compute the minimal model J of the definite Horn program P^I;
    3. check whether $I = J$.
  - Hardness is shown via a simple reduction $\mathcal{T}$ from SAT to normal logic programs, such that, for each formula ϕ it holds, that ϕ is satisfiable iff $\mathcal{T}[\phi]$ has a stable model (blackboard!).
- Alternative proof: Via an encoding of an NTM.

Disjunctive Programs—Introduction

- Idea: Add disjunctions to the heads.
- Makes the formulation of the "generate"-part easier.
- Example: 3-coloring of graphs; defined as 2-coloring but now with 3 colors, say red, green, and blue.
  - Let a set of facts $e_{i,j}$ specify our graph over vertices $v_1, \ldots, v_n$.
  - Generate Part:
$$\{r_i \vee g_i \vee b_i \leftarrow \mid 1 \leq i \leq n\}.$$
  - Check Part:
$$\begin{aligned} \{ \quad & \bot \leftarrow e_{i,j}, r_i, r_j; \\ & \bot \leftarrow e_{i,j}, g_i, g_j; \\ & \bot \leftarrow e_{i,j}, b_i, b_j \mid 1 \leq i \leq n; 1 \leq j \leq n\}. \end{aligned}$$

Disjunctive Programs—Syntax

- A disjunctive rule r (over $\mathcal{A}$) is an expression of the form

$$h_1 \vee \cdots \vee h_k \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m$$

  where $h_1, \ldots, h_k, b_1, \ldots, b_m$ are propositional atoms (from $\mathcal{A}$), and $k \geq 0$, $n \geq 0$.

- We call
  - $H(r) = \{h_1, \ldots, h_k\}$ the *head* of r;
  - $B(r) = \{b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m\}$ the *body* of r;
  - $B^+(r) = \{b_1, \ldots, b_n\}$ the *positive body* of r;
  - $B^-(r) = \{b_{n+1}, \ldots, b_m\}$ the *negative body* of r.

- A *disjunctive program* is a set of disjunctive rules.

Disjunctive Programs—Semantics

- Let r be a rule

$$h_1 \vee \cdots \vee h_k \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m$$

An interpretation $I \subseteq \mathcal{A}$ is a *model* of r iff the following holds:

If $b_1, \ldots, b_n$ are all in I, and none of $b_{n+1}, \ldots, b_m$ are in I then at least one out of $h_1, \ldots, h_k$ is in I.

- Define for r as above:

$$\hat{r} \;=\; b_1 \wedge \cdots \wedge b_n \wedge \neg b_{n+1} \wedge \cdots \wedge \neg b_m \;\supset\; h_1 \vee \cdots \vee h_k.$$

  - Then I is a model of a rule r iff I is a model of the formula $\hat{r}$.

- An interpretation $I \subseteq \mathcal{A}$ is a *model* of a disjunctive program P iff I is a model of each $r \in P$.

  - As before: I is a model of a program P iff I is a model of the *associated theory* $\hat{P} = \{\hat{r} \mid r \in P\}$.

- Again, $I \models r$ (resp. $I \models P$) denotes that I is model of r (resp. P).

Disjunctive Programs—Semantics (ctd.)

- As before, we define the *reduct* of a disjunctive program P with respect to some interpretation I:

$$P^I \;=\; \{H(r) \leftarrow B^+(r) \mid r \in P; I \cap B^-(r) = \emptyset; \}$$

  ☞ The reduct is no longer a Horn-program!

- Let I be an interpretation; P a disjunctive program. Then, I is a *stable model* (or an *answer set*) of P iff I is a minimal model of P^I.

- Observation: For disjunctive programs without negation (positive programs) the stable models of a program coincide with the minimal classical models of its associated theory.

Disjunctive Programs—Semantics (ctd.)

- Is there any difference between disjunction and using negation?
- Observe:

$$P = \{p \vee q \leftarrow\} \qquad Q = \{p \leftarrow not\ q; \\ q \leftarrow not\ p\}$$

  share the same stable models $\{p\}$ and $\{q\}$.

- But adding a *cycle*

$$P = \{ \quad p \vee q; \leftarrow \qquad Q = \{ \quad p \leftarrow not\ q; \\ q \leftarrow not\ p; \\ p \leftarrow q; \qquad p \leftarrow q; \\ q \leftarrow p\ \} \qquad q \leftarrow p\ \}$$

  yields $\{p, q\}$ is stable model of P, but Q has no stable model.

Disjunctive Programs—Semantics (ctd.)

- Some observations:
  - For any interpretation I and any disjunctive program P, $I \models P$ iff $I \models P^I$.
  - **Proposition.** Let P be a disjunctive program and I an interpretation. Then, I is a stable model of P iff
    - $I \models P$; and
    - for each $J \subset I$, $J \not\models P^I$.
  - For each disjunctive programs P, and each pair I, J of stable models of P, $I \subseteq J$ implies $I = J$; i.e., the stable models of any program are pairwise *incomparable*.

Disjunctive Programs—Complexity

- Given a disj. program P, and an interpretation I; checking whether I is a stable model of P is in coNP (in fact, it is complete for coNP):
  - Check $I \models P$ and UNSAT for the theory associated to

$$P^I \cup \{\bot \leftarrow I\} \cup \{\bot \leftarrow a \mid a \in \mathcal{A} \setminus I\}.$$

- Given a disj. program P, checking whether P has at least one stable model is Σ_2^P-complete [Eiter & Gottlob; Ann. Math. Artif. Intell. 95]
  - Membership:
    1. Guess an interpretation I;
    2. check whether I is stable model of P^I; (in coNP).
  - Hardness is shown via a reduction $\mathcal{T}$ from $(2, \exists)$-QSAT to disjunctive programs, such that for each $(2, \exists)$-QBF Φ, Φ is true iff $\mathcal{T}[\Phi]$ has a stable model (blackboard!).

Program Classes

- So far, we introduced the following program classes over rules

$$h_1 \vee \cdots \vee h_k \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m.$$

A program P is called

definite Horn		$k = 1;\ m = n;$
Horn	iff	$k \leq 1; m = n;$
normal	for each $r \in P$	$k \leq 1;$
definite		$k \geq 1; m = n;$
positive		$m = n;$
disjunctive		(no restriction).

Program Classes (ctd.)

- Further classes may be defined using the *dependency graph* of a program P.
- $D(P)$ is given as follows, having two kinds of edges E^+, E^-:
  - the vertices V are the propositional atoms in P;
  - there is an edge in E^+ from p to q, iff there is a rule $r \in P$, such that $p \in H(r)$, $q \in B^+(r)$;
  - there is an edge in E^- from p to q, iff there is a rule $r \in P$, such that $p \in H(r)$, $q \in B^-(r)$.
- Identify $D^+(P) = (V, E^+)$, $D^-(P) = (V, E^-)$, and $D(P) = (V, E^+ \cup E^-)$.

Program Classes (ctd.)

- A logic program P is called
  - *stratified* iff each cycle in $D(P)$ has its edges only from E^+.
  - *acyclic* iff $D^+(P)$ contains no cycle.
  - *head-cycle free* (HCF) iff there is no cycle in $D^+(P)$ going through two distinct atoms from a head $H(r)$, $r \in P$.
- Examples:
  - $\{p \leftarrow not\ q\}$ is stratified, acyclic, HCF;
  - $\{p \leftarrow not\ q;\ q \leftarrow not\ p\}$ is not stratified, acyclic, HCF;
  - $\{p \vee q \leftarrow\}$ is stratified, acyclic, HCF;
  - $\{p \leftarrow q;\ q \leftarrow p\}$ is stratified, not acyclic ,and HCF;
  - $\{p \vee q \leftarrow;\ p \leftarrow q;\ q \leftarrow p\}$ is stratified, not acyclic, not HCF.

Program Classes (ctd.)

- Further generalizations of programs:
  - *Classical negation*; programs are not given over atoms but over classical literals; (this is more or less syntactic sugar);
  - *Nested logic programs* have rules of the form:

    $$H \leftarrow B$$

    where H and B are arbitrary expressions built from atoms using "$\vee$", "$,$", and "not".
  - *Equilibrium logic* [Pearce 99], provides answer-set like semantics (equilibrium models) for propositional theories; if the theory T is associated to some program P, the equilibrium models of T are in 1-1 correspondence to the answer sets of P.

Program Classes (ctd.)

- In practice, logic programs are often enriched by different features:
  - cardinality (weight) constraints: atoms are considered as expressions
  $$n\{a_1, \ldots, a_n\}m$$
  which are true under I iff $n \leq |I \cap \{a_1, \ldots, a_n\}| \leq m$ holds;
  - built-in predicates (e.g., arithmetic predicates);
  - weak constraints for optimization problems (e.g., TSP);
  - aggregates (similarly as used in databases).

Relation to Classical Models

- We already defined the notions of models of programs, by considering the associated theory.
- How to talk about reducts in classical logic?
- Solution: Use renaming!
  - For a rule r of the form

$$h_1 \vee \cdots \vee h_k \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m.$$

  - We define

$$r^* = b'_1 \wedge \cdots \wedge b'_n \wedge \neg b_{n+1} \wedge \cdots \wedge \neg b_m \supset h'_1 \vee \cdots \vee h'_k.$$

    Hence, $H(r^*) = H(r')$, $B^+(r^*) = B(r')$, and $B^-(r^*) = B^-(r)$.
  - Moreover, define

$$P^* = \{r^* \mid r \in P\}.$$

Relation to Classical Models (ctd.)

- Example: For $P = \{a \vee b \leftarrow c, not\ d\}$, we obtain $P^* = \{c' \wedge \neg d \supset (a' \vee b')\}$.

  We have that any interpretation I with $d \in I$ is a model of P^*. Moreover, for instance, $\{c', c, a', a\}$ is model of P^*, etc.

- **Proposition.** Let P be a program over atoms V; let $J, K \subseteq V$; and let I be any interpretation, such that $(I \cap V) = J$ and $(I \cap V') = K'$. Then,

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

- Now we can use P^* to compute stable models via QBFs.

Relation to Classical Models (ctd.)

- For program classes which are located in NP, efficient reductions to propositional formulas are possible
  - For acyclic (or "tight") programs, *program completion* is sufficient [Erdem & Lifschitz, TPLP 2003]
  - For HCF programs, encodings make use of level mappings [Ben-Eliyahu & Dechter, Ann. Math. Artif. Intell. 1994]
- Encodings to propositional logic are always possible, if we take an exponential blow-up in the worst case
  - Central concept: *Loop formulas* [Lin & Zhao, AIJ 2004; Ferraris, Lee, Lifschitz, Ann. Math. Artif. Intell. 2006]

Exercises

- Show that for any program, the stable models are pairwise incomparable. Hint: First, show that $I \models P$ implies $I \models P^J$ for any $I \subseteq J$.
- Construct a function $\mathcal{T}$ mapping any disjunctive program P over atoms V to an open QBF $\mathcal{T}[P]$ over atoms $V \cup V'$ (with atoms from V being free) such that the models of $\mathcal{T}[P]$ match the stable models of P.