**Translatability and Intranslatability Results for Certain Classes of Logic Programs**

Tomi Janhunen  
Helsinki University of Technology  
Department of Computer Science and Engineering  
Laboratory for Theoretical Computer Science  
December 4, 2003

---

**BACKGROUND AND MOTIVATION**

*Example:* Suppose that $P$ contains a rule $a \leftarrow b_1, \ldots, b_n$ and the head $a$ is known to be false in a model $M$ of $P$ being constructed.

$\implies$ One of $b_1, \ldots, b_n$ must be false in $M$ (if $M \models P$ is to hold).

1. If $n = 1$, then we know immediately that $b_1$ is false in $M$.
2. If, in addition, $b_1, \ldots, b_{i-1}$ and $b_{i+1}, \ldots, b_n$ are known to be true in $M$, then $b_i$ is false in $M$.

Q: Can we somehow reduce the number of positive subgoals in rules?

---

**1. PRELIMINARIES: NORMAL PROGRAMS**

* A normal (logic) program $P$ is a set of rules of the form $a \leftarrow b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m$.

* We use the following notations for a rule $r$ of the kind above:

  $\text{head}(r) = a$,

  $\text{body}^+(r) = \{b_1, \ldots, b_n\}$, $\text{body}^-(r) = \{c_1, \ldots, c_m\}$, and

  $\text{body}(r) = \{b_1, \ldots, b_n, \neg c_1, \ldots, \neg c_m\}$.

* A rule $r \in P$ is satisfied in a propositional interpretation $I \subseteq \text{Hb}(P)$ if $I \models \text{body}(r)$ implies $I \models \text{head}(r)$.

* An interpretation $M \subseteq \text{Hb}(P)$ is a (classical) model of $P$ (denoted by $M \models P$) if $M \models r$ holds for all $r \in P$.
Minimal Models

Definition: A model $M \models P$ is a minimal model of $P$
$\iff$ there is no model $M' \models P$ such that $M' \subseteq M$.

- Every positive (negation free) normal program $P$ has a unique minimal model $LM(P)$, i.e., the least model of $P$.
- The least model $LM(P) = \text{lfp}(T_P)$ where $T_P$ is an operator defined by $T_P(A) = \{ \text{head}(r) \mid r \in P \text{ and } body^+(r) \subseteq A \}$.
- Given $a \in LM(P) = \text{lfp}(T_P)$, the level number $\#a$ is the least number $i > 0$ such that $a \in T_P \upharpoonright i$ but $a \notin T_P \upharpoonright i - 1$.

Example: For $P = \{ a \leftarrow; b \leftarrow a; c \leftarrow b; a \leftarrow b; d \leftarrow d \}$, the least model $LM(P) = \{ a, b, c \}$.
The respective level numbers are $\#a = 1$, $\#b = 2$, and $\#c = 3$.

Support Models

- Given an interpretation $M \subseteq \text{Hb}(P)$, we define the set of supporting rules
  $SR(P, M) = \{ r \in P \mid M \models \text{body}(r) \}$.

Definition: An interpretation $M \subseteq \text{Hb}(P)$ is a supported model of $P$ $\iff$ $M \models P$ and $\forall a \in M: \exists r \in SR(P, M)$ such that $\text{head}(r) = a$.

Example: A positive program $P = \{ a \leftarrow b; b \leftarrow a \}$ has two supported models $M_1 = \emptyset$ and $M_2 = \{ a, b \}$, but only $M_1$ is stable.

Proposition: Stable models of $P$ are also supported models of $P$ (but the converse does not hold in general).

Stable Models

- Given an interpretation $M \subseteq \text{Hb}(P)$, the Gelfond-Lifschitz reduct $P^M = \{ \text{head}(r) \leftarrow \text{body}^+(r) \mid r \in P \text{ and } \text{body}^-(r) \cap M = \emptyset \}$.

Definition: An interpretation $M \subseteq \text{Hb}(P)$ of a normal logic program $P$ is a stable model of $P$ $\iff M = LM(P^M)$.

Example: A program $P = \{ a \leftarrow \neg b \}$ has three classical models $M_1 = \{ a \}$, $M_2 = \{ b \}$, and $M_3 = \{ a, b \}$, but only $M_1$ is stable:
$P^{M_1} = \{ a \leftarrow \}$ and $P^{M_2} = P^{M_3} = \emptyset$.

Proposition: Stable models of $P$ are also classical models of $P$ (but the converse does not hold in general).

2. General Assumptions on Logic Programs

Definition: A logic program is a triple $(P, A, V)$ where
1. $P$ is a set of expressions (such as rules, clauses or sentences) built of propositional atoms;
2. $A$ is a set of additional atoms that need not appear in $P$; and
3. $V$ defines which atoms appearing in $P$ and $A$ are visible.

By a slight abuse of notation, we write $P$ for $(P, A, V)$, $\text{Hb}_A(P)$ for $A$, $\text{Hb}(P)$ for the set of atoms appearing in $P$ and $A$, and $\text{Hb}_v(P)$ for $V$.
The hidden part $\text{Hb}_h(P)$ is $\text{Hb}(P) - \text{Hb}_v(P)$.
Unless otherwise stated, $\text{Hb}_A(P) = \emptyset$ and $\text{Hb}_v(P) = \text{Hb}(P)$.

Example: A logic program $P = \{ a \leftarrow \neg a \}$ with $\text{Hb}(P) = \{ a, b \}$ and $\text{Hb}_v(P) = \{ a \}$ has two classical models $M_1 = \{ a \}$ and $M_2 = \{ a, b \}$.
Requirements for Classes of Logic Programs

Each class of logic programs $\mathcal{C}$ must satisfy the following criteria:
1. Each member $P \in \mathcal{C}$ is a finite set of expressions and the Herbrand base $\text{Hb}(P)$ is finite.
2. Closure under unions: if $P \in \mathcal{C}$ and $Q \in \mathcal{C}$, then $P \cup Q \in \mathcal{C}$.
3. Closure under intersections: if $P \in \mathcal{C}$ and $Q \in \mathcal{C}$, then $P \cap Q \in \mathcal{C}$.
4. There is a semantical operator $\text{Sem}_\mathcal{C}$ that maps a program $P \in \mathcal{C}$ to a set of models $\text{Sem}_\mathcal{C}(P) \subseteq 2^{\text{Hb}(P)}$, i.e., the set of models of $P$.

Example: The class of finite normal programs $\mathcal{P}$ satisfies these criteria but $\mathcal{P}_{\text{odd}} = \{ P \in \mathcal{P} \mid P \text{ has an odd number of rules} \}$ does not.

Example: Some Syntactic Subclasses of $\mathcal{P}$

- By constraining the number of positive body literals $n$, we obtain the following subclasses of normal programs:
  1. The class of atomic programs $\mathcal{A}$ ($n = 0$ for every rule).
  2. The class of unary programs $\mathcal{U}$ ($n \leq 1$ for every rule).
  3. The class of binary programs $\mathcal{B}$ ($n \leq 2$ for every rule).
  $\mathcal{A} \subseteq \mathcal{U} \subseteq \mathcal{B} \subseteq \mathcal{P}$.
- For each class $\mathcal{C} \in \{ \mathcal{A}, \mathcal{U}, \mathcal{B}, \mathcal{P} \}$, the semantics is determined by $\text{Sem}_\mathcal{C}(P) = \text{SM}(P) = \{ M \subseteq \text{Hb}(P) \mid M \models \text{L}(P^M) \}$.
- The classes of positive programs $\mathcal{A}^+ \subseteq \mathcal{U}^+ \subseteq \mathcal{B}^+ \subseteq \mathcal{P}^+$ are obtained analogously by denying negative body literals.

Example: Sets of Clauses

- In analogy to rules, propositional clauses $a_1 \lor \cdots \lor a_n \lor \neg b_1 \lor \cdots \lor \neg b_m$ are expressions formed of propositional atoms.
- We write $\mathcal{SC}$ for the class of finite sets of clauses.
- The semantics of a set $S \in \mathcal{SC}$ is determined by an operator $\text{Sem}_\mathcal{SC}(S) = \text{CM}(S) = \{ M \subseteq \text{Hb}(S) \mid M \models S \}$.

$\mathcal{SC}$ can be viewed as a class of logic programs.

3. Notion of Equivalence

Definition: Logic programs $P \in \mathcal{C}$ and $Q \in \mathcal{C}'$ are visibly equivalent (denoted by $P \equiv_v Q$) iff
1. $\text{Hb}_v(P) = \text{Hb}_v(Q)$ and
2. there is a bijective function $f : \text{Sem}_\mathcal{C}(P) \rightarrow \text{Sem}_{\mathcal{C}'}(Q)$ such that $M \cap \text{Hb}_v(P) = f(M) \cap \text{Hb}_v(Q)$ holds for every $M \in \text{Sem}_\mathcal{C}(P)$.

This notion is applicable both within a single class of programs as well as between different classes of programs.

The number of models is preserved under $\equiv_v$. 
Example:

1. The stable models of a normal logic program

   \[ P = \{ a \leftarrow \neg b; \ b \leftarrow \neg a; \ c \leftarrow a; \ c \leftarrow \neg a \} \]

   with \( \text{Hb}(P) = \{ a, b, c \} \) are \( M_1 = \{ a, c \} \) and \( M_2 = \{ b, c \} \).

2. The set of clauses

   \[ S = \{ a \lor d, \ \neg a \lor \neg d, \ a \lor c, \ \neg a \lor c \} \]

   has exactly two classical models \( N_1 = \{ a, c \} \) and \( N_2 = \{ d, c \} \),

   since \( \text{Hb}(S) = \{ a, c \} \).

   The atoms \( b \) and \( d \), which appear in \( P \) and \( S \), respectively, can be

   hidden by tuning the visibility of atoms: \( \text{Hb}_v(P) = \text{Hb}_v(S) = \{ a, c \} \).

   \[ P \equiv_v S \text{ holds.} \]

4. Properties of Translation Functions

   Definition: A translation function \( Tr : C \rightarrow C' \) is polynomial \((P)\)

   \[ \iff \text{for all } P \in C, \text{ the time required to compute } Tr(P) \text{ is polynomial} \]

   in \( ||P|| \), i.e., the number of symbols needed to represent \( P \).

   Definition: A translation function \( Tr : C \rightarrow C' \) is faithful \((F)\)

   \[ \iff \text{for all } P \in C, \ P \equiv_v Tr(P). \]

   Example: Consider a hypothetical translation function

   \[ Tr_{\text{double}}(P) = P \cup \{ a \leftarrow \neg b; \ b \leftarrow \neg a \}, \]

   where \( a \notin \text{Hb}(P) \) and \( b \notin \text{Hb}(P) \) are two new atoms, and

   \[ \text{Hb}_v(Tr_{\text{double}}(P)) = \text{Hb}_v(P) \text{ by definition.} \]

   \[ \text{Tr}_{\text{double}} \text{ is linear (and thus polynomial) but not faithful.} \]

Properties of Translation Functions (continued)

Module conditions for two programs \( P \in C \) and \( Q \in C \) are:

1. \( P \cap Q = \emptyset \) \quad 2. \( \text{Hb}_a(P) \cap \text{Hb}_a(Q) = \emptyset \)
2. \( \text{Hb}_b(P) \cap \text{Hb}_b(Q) = \emptyset \) \quad 4. \( \text{Hb}(P) \cap \text{Hb}(Q) = \emptyset \)

Definition: A translation function \( Tr : C \rightarrow C' \) is modular \((M)\)

\[ \iff \text{for all } P \in C \text{ and } Q \in C \text{ satisfying } M1-M4, \]

\( Tr(P \cup Q) = Tr(P) \cup Tr(Q); \) and \( Tr(P) \text{ and } Tr(Q) \) satisfy \( M1-M4. \)

Definition: A translation function \( Tr : C \rightarrow C' \) is PFM

\[ \iff Tr \text{ is polynomial, faithful, and modular.} \]

Proposition: Any composition of polynomial/faithful/modular

translation functions is also polynomial/faithful/modular.
Classification Method

Given two classes $C$ and $C'$ of programs, the goal is to establish either

$\triangleright$ $C \subseteq_{\text{PFM}} C'$ (there is a PFM translation function $\text{Tr} : C \rightarrow C'$), or
$\triangleright$ $C \not\subseteq_{\text{PFM}} C'$ (such a translation function does not exist).

These relations induce further relations for classes of logic programs:

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C &lt;_{\text{PFM}} C'$</td>
<td>$C \subseteq_{\text{PFM}} C'$ and $C' \not\subseteq_{\text{PFM}} C$</td>
<td>strictly less</td>
</tr>
<tr>
<td>$C =_{\text{PFM}} C'$</td>
<td>$C \subseteq_{\text{PFM}} C'$ and $C' \subseteq_{\text{PFM}} C$</td>
<td>equal</td>
</tr>
<tr>
<td>$C \neq_{\text{PFM}} C'$</td>
<td>$C \not\subseteq_{\text{PFM}} C'$ and $C' \not\subseteq_{\text{PFM}} C$</td>
<td>incomparable</td>
</tr>
</tbody>
</table>

Classes can be ordered on the basis of their expressive power.

5. EXPRESSION POWER ANALYSIS

Proposition: The inclusions $A^+ \subseteq U^+ \subseteq B^+ \subseteq P^+$ imply $A^+ \subseteq_{\text{PFM}} U^+ \subseteq_{\text{PFM}} B^+ \subseteq_{\text{PFM}} P^+$.

Theorem: $U^+ \not\subseteq_{\text{PFM}} A^+$

Proof. Suppose that $\text{Tr} : U^+ \rightarrow A^+$ is faithful and modular.

Programs $P = \{a \leftrightarrow b\}$ and $Q = \{b \leftrightarrow\}$ satisfy M1-M4.

Thus $\text{Tr}(P \cup Q) = \text{Tr}(P) \cup \text{Tr}(Q)$ which are disjoint and atomic.

Now $a \in \text{LM}(P \cup Q) \implies a \in \text{LM}(\text{Tr}(P) \cup \text{Tr}(Q))$

$\implies a \in \text{LM}(\text{Tr}(P))$ or $a \in \text{LM}(\text{Tr}(Q))$

$\implies a \in \text{LM}(P)$ or $a \in \text{LM}(Q)$, a contradiction.

Summary of our Results for Normal Programs

$\triangleright$ $A \subseteq U \subseteq B \subseteq P \implies A \subseteq_{\text{PFM}} U \subseteq_{\text{PFM}} B \subseteq_{\text{PFM}} P$.

$\triangleright$ Non-binary rules can be translated away: $P \not\subseteq_{\text{PFM}} B$.

$\triangleright$ Binary and unary rules cannot be translated away in a faithful and modular way: $B \not\subseteq_{\text{PFM}} U$ and $U \not\subseteq_{\text{PFM}} A$.

$\triangleright$ It is straightforward to encode propositional satisfiability problems as (atomic) normal programs: $SC\subseteq_{\text{PFM}} A$.

$\triangleright$ Due to non-monotonicity $A \not\subseteq_{\text{PFM}} SC$.

$\triangleright$ The resulting hierarchy of the five classes under consideration:

$SC \subseteq_{\text{PFM}} A \subseteq_{\text{PFM}} U \subseteq_{\text{PFM}} B =_{\text{PFM}} P$. 
6. YET ANOTHER CHARACTERIZATION OF STABILITY

Definition: Given a supported model $M$ of $P$, a function $\#$ from $M \cup \text{SR}(P,M)$ to $\mathbb{Z}^+$ is a level numbering w.r.t. $M \iff$

1. $\forall a \in M$: $\#a = \min \{\#r \mid r \in \text{SR}(P,M) \text{ and } a = \text{head}(r)\}$ and $\forall r \in \text{SR}(P,M)$:

$$\#r = \begin{cases} \max \{\#b \mid b \in \text{body}^+(r)\} + 1, & \text{if } \text{body}^+(r) \neq \emptyset. \\ 1, & \text{otherwise.} \end{cases}$$

In addition, atoms, level numbers are assigned to rules.

Characterization of Stability (Continued)

Theorem: If $M$ is a stable model of $P$, then $M$ is a supported model of $P$ and there exists a unique level numbering $\#$ w.r.t. $M$:

1. For $a \in M$, $\#a$ is defined as for the members of $\text{Iff}(P_{\text{sup}})$.
2. For $r \in \text{SR}(P,M)$, $\#r = \max\{|1\} \cup \{\#b + 1 \mid b \in \text{body}^+(r)\}$.

If $M$ is a supported model of $P$ and there is a level numbering $\#$ w.r.t. $M$, then $\#$ is unique and $M$ is a stable model of $P$.

Example: Recall $P = \{r_1, r_2\}$, where $r_1 = a \leftarrow b$ and $r_2 = b \leftarrow a$, and the second supported model $M = \{a, b\}$ of $P$.

The requirements for a level numbering w.r.t. $M$ lead to four equations: $\#a = \#r_1$, $\#r_1 = \#b + 1$, $\#b = \#r_2$, and $\#r_2 = \#a + 1$.

There is no solution $\implies M$ is not stable.

7. NON-MODULAR TRANSLATION FUNCTIONS

- Despite the preceding intranslatability results we will seek for polynomial, faithful and non-modular (PF) translation functions.
- The first goal is to translate any normal program $P$ into an atomic one $\text{T}_{\text{AT}}(P)$.
- The preceding characterization of stable models suggests a translation that consists of two fairly independent parts:
  1. The first part captures a supported model $M$ of $P$.
  2. The second part checks if one can assign a level numbering (as described above) for atoms $a \in M$ and rules $r \in \text{SR}(P,M)$.
- The result is to be a polynomial and faithful translation function such that $||\text{T}_x(P)||$ is of order $||P|| \times \log_2|\text{Hb}(P)|$.

Capturing Supported Models: $\text{T}_{\text{sup}}(P)$

- The complementary atom $\bar{a}$ is defined for each $a \in \text{Hb}(P)$:

$$\bar{a} \leftarrow \sim a.$$

- A rule $r \in P$ is translated as follows:

$$\bar{bt}(r) \leftarrow \sim \text{body}^+(r), \sim \text{body}^-(r),$$

$$\bar{bt}(r) \leftarrow \sim bt(r), \text{ and}$$

$$\text{head}(r) \leftarrow \sim \text{bt}(r)$$

where $bt(r)$ is a new atom denoting that “the body of $r$ is true”.

- New atoms are necessary here in order to avoid quadratic blow up in the rest of the translation.
Binary Counters: $\text{Tr}_{\text{CTR}}(P)$

- The number of bits $\nabla P = \lceil \log_2(|\text{Hb}(P)| + 2) \rceil$.
- We introduce a binary counter (two vectors of atoms)
  $\text{ctr}(a) = \text{ctr}(a)_1 \ldots \text{ctr}(a)_{\nabla P}$ and $\overline{\text{ctr}(a)} = \overline{\text{ctr}(a)}_1 \ldots \overline{\text{ctr}(a)}_{\nabla P}$
  for each $a \in \text{Hb}(P)$.
- The value of $\text{ctr}(a)$ is chosen if $a \in M$, i.e., $\overline{a}$ cannot be derived:
  a subprogram $\text{SEL}_{\nabla P}(\text{ctr}(a), \overline{a})$ does the job.
- Similarly, we need to define another counter $\text{nxt}(a)$ that takes the value
  of $\text{ctr}(a)$ incremented by one: $\text{NXT}_{\nabla P}(\text{ctr}(a), \text{nxt}(a), \overline{a})$.
- For $r \in P$ with $\text{body}^+(r) \neq \emptyset$, we need $\text{SEL}_{\nabla P}(\text{ctr}(r), \text{bt}(r))$.
- For $r \in P$ with $\text{body}^+(r) = \emptyset$, $\text{FIX}_{\nabla P}(\text{ctr}(r), 1, \text{bt}(r))$ is enough.

Checking Minimality: $\text{Tr}_{\text{MIN}}(P)$

- The value of $\text{ctr}(a)$ is supposed to be $\#a$ in binary.
- For each rule $r$ and $a = \text{head}(r)$, we need the subprograms
  $\text{LT}_{\nabla P}(\text{ctr}(r), \text{ctr}(a), \text{bt}(r))$ and $\text{EQ}_{\nabla P}(\text{ctr}(r), \text{ctr}(a), \text{bt}(r))$
in addition to the following rules:
  $y \leftarrow \neg y, \neg \text{bt}(r), \neg \text{lt}(\text{ctr}(r), \text{ctr}(a))$; and
  $\text{min}(a) \leftarrow \neg \text{bt}(r), \neg \text{eq}(\text{ctr}(r), \text{ctr}(a))$.
- For each $a \in \text{Hb}(P)$, we introduce the rule $y \leftarrow \neg y, \neg \overline{a}, \neg \text{min}(a)$.

Checking Maximaly: $\text{Tr}_{\text{MAX}}(P)$

- The value of $\text{ctr}(r)$ is supposed to be $\#r$ in binary.
- If $\text{body}^+(r) \neq \emptyset$, we need for each $b \in \text{body}^+(r)$ subprograms
  $\text{LT}_{\nabla P}(\text{ctr}(r), \text{nxt}(b), \text{bt}(r))$ and $\text{EQ}_{\nabla P}(\text{ctr}(r), \text{nxt}(b), \text{bt}(r))$
  plus the following rules:
  $x \leftarrow \neg x, \neg \text{bt}(r), \neg \text{lt}(\text{ctr}(r), \text{nxt}(b))$;
  $\text{max}(r) \leftarrow \neg \text{bt}(r), \neg \text{eq}(\text{ctr}(r), \text{nxt}(b))$; and
  $x \leftarrow \neg x, \neg \text{bt}(r), \neg \text{max}(r)$.
- The case that $\text{body}^+(r) = \emptyset$ is handled by $\text{Tr}_{\text{CTR}}(P)$.

Non-Modular Translation Functions (Continued)

- The next objective is to embed $A$ into $\mathcal{S}_C$.
  **Definition:** For an atomic normal program $P \in A$ and an atom $a \in \text{Hb}(P)$, let $\text{Def}_P(a) = \{r \in P \mid \text{head}(r) = a\}$,
  $\text{Tr}_{\text{CL}}(a, P) = \{a \lor \neg \text{bt}(r) \mid a \in \text{Hb}(P) \text{ and } r \in \text{Def}_P(a)\} \cup \{\neg a \lor \bigvee \{\text{bt}(r) \mid r \in \text{Def}_P(a)\} \mid a \in \text{Hb}(P)\} \cup \{\text{bt}(r) \lor \bigvee \text{body}^-(r) \mid r \in \text{Def}_P(a)\} \cup \{\neg \text{bt}(r) \lor \neg c \mid r \in \text{Def}_P(a) \text{ and } c \in \text{body}^-(r)\}$,
  and $\text{Tr}_{\text{CL}}(P) = \bigcup_{a \in \text{Hb}(P)} \text{Tr}_{\text{CL}}(a, P)$.

$A \subseteq_{\text{PF}} \mathcal{S}_C, P \subseteq_{\text{PF}} \mathcal{S}_C$, and $\mathcal{S}_C =_{\text{PF}} A =_{\text{PF}} U =_{\text{PF}} B =_{\text{PF}} P$, respectively.
8. RELATED WORK

- I. Niemelä [AMAI, 1999]: *Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm.*
  - A counter-example which shows that normal programs cannot be translated into sets of clauses in a faithful and modular way.
- Capturing propositional satisfiability with normal programs.
- S. Brass and J. Dix [JLP, 1999]: *Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation.*
  - Example: In partial evaluation, a rule $a \leftarrow b, \neg c$ is replaced by $a \leftarrow \neg b_1, \neg c$ and $a \leftarrow \neg b_1, c$ if the definition of $b$ consists of $b \leftarrow \neg b_1$ and $b \leftarrow \neg b_2$.
  - An exponential space is needed in the worst case.

---

Related Work (Continued)

- G. Antoniou et al. [ACM TOCL, 2001]: *Representation Results for Defeasible Logic.*
  - They study transformations on a class of defeasible theories:
    1. Correctness: $D \equiv_{L(D)} \text{Tr}(D)$.
    2. Incrementality: $D_1 \cup D_2 \equiv_{L(D_1) \cup L(D_2)} \text{Tr}(D_1) \cup \text{Tr}(D_2)$.
    3. Modularity: $D_1 \cup D_2 \equiv_{L(D_1) \cup L(D_2)} D_1 \cup \text{Tr}(D_2)$.
  - The semantics of defeasible theories is quite different.
  - The notion of correctness is close to our notion of faithfulness.
  - The other two conditions are semantic rather than syntactic.

---

Related Work (Continued)

- R. Ben-Eliyahu and R. Dechter [AMAI, 1994]: *Propositional semantics for disjunctive logic programs.*
  - Binary numbers are not used $\iff$ at least quadratic encoding.
  - The stable models of $P$ and the classical models of $\text{Tr}_{ED}(P)$ are not in a bijective relationship.
  - Example: Let $P = \{a \leftarrow b, c; b \leftarrow d; c \leftarrow d; d \leftarrow e; d \leftarrow a\}$.
    - The atoms $b$ and $c$ in the unique stable model $M = \{a, b, c, d\}$ can be ordered in two different ways (in a total ordering).
  - Y. Babovich, E. Erdem, and V. Lifschitz [NMR Workshop, 2000]: *Fages’ Theorem and Answer Set Programming.*
    - Programs containing loops are not (necessarily) covered.
  - Tightness is based on a different numbering of atoms.

---

Related Work (Continued)

  - In this approach, disjunctive stable models are captured with quantified Boolean formulas $\exists p_1 \ldots \exists p_n \forall q_1 \ldots \forall q_m \phi$.
  - In particular, the minimality requirement of disjunctive stable models is easy to express using such a formula.
  - From the point of view of complexity, the computation of stable models is easier in the case of normal logic programs.
  - The idea is to extend the completion of $P$ [Clark, 1978] with loop formulas to exclude non-stable models.
  - In the worst case, there is an exponential number of loops (for instance, Hamiltonian paths for complete graphs).
9. CONCLUSIONS

- It is not easy to remove all positive body literals.
- EPH (PFM): SC <_{PFM} A <_{PFM} U <_{PFM} B =_{PFM} P.
- EPH (PF): SC =_{PF} A =_{PF} U =_{PF} B =_{PF} P.
- Distinctive features of the counter based approach:
  1. bijective relationship of models and
  2. ||Tr(P)|| is of order ||P|| \times \log_2 |Hb(P)|.
- Transitive closure can be properly captured with classical models.
- Experimental results with the implementations of Tr_{AT} and Tr_{CL}
  are promising, but further optimizations should be pursued for in
  order to really compete with SMODELS.