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

OUTLINE OF THE TALK

1. Preliminaries: Normal Programs
2. General Assumptions on Logic Programs
3. Notion of Equivalence
4. Properties of Translation Functions
5. Expressive Power Analysis
6. Yet Another Characterization of Stability
7. Non-Modular Alternatives
8. Related Work
9. Conclusions

BACKGROUND AND MOTIVATION

Example: Suppose that P contains a rule $\mathsf{a} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_n$ and the head a is known to be false in a model M of P being constructed.

$\Longrightarrow$ One of $\mathsf{b}_1, \ldots, \mathsf{b}_n$ 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, $\mathsf{b}_1, \ldots, \mathsf{b}_{i-1}$ and $\mathsf{b}_i, \ldots, \mathsf{b}_n$ 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?

T. Janhunen [CL 2000]: *Comparing the Expressive Powers of Some Syntactically Restricted Classes of Logic Programs.*

T. Janhunen [ASP, 2003]: *A Counter-Based Approach to Translating Logic Programs into Sets of Clauses.*

1. PRELIMINARIES: NORMAL PROGRAMS

➢ A **normal (logic) program** P is a set of **rules** of the form

$$\mathsf{a} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_n, \sim\mathsf{c}_1, \ldots, \sim\mathsf{c}_m.$$

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

$\mathrm{head}(r) = \mathsf{a}$,
$\mathrm{body}^+(r) = \{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$, $\mathrm{body}^-(r) = \{\mathsf{c}_1, \ldots, \mathsf{c}_m\}$, and
$\mathrm{body}(r) = \{\mathsf{b}_1, \ldots, \mathsf{b}_n, \sim\mathsf{c}_1, \ldots, \sim\mathsf{c}_m\}$.

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

➢ An interpretation $M \subseteq \mathrm{Hb}(P)$ is a **(classical) model** of P (denoted by $M \models P$) $\iff$ $M \models r$ 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* $M' \subset M$.

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

Example: For $P = \{\mathsf{a} \leftarrow;\ \mathsf{b} \leftarrow \mathsf{a};\ \mathsf{c} \leftarrow \mathsf{b};\ \mathsf{a} \leftarrow \mathsf{b};\ \mathsf{d} \leftarrow \mathsf{d}\}$, the least model $\mathrm{LM}(P) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$.

The respective level numbers are $\#\mathsf{a} = 1$, $\#\mathsf{b} = 2$, and $\#\mathsf{c} = 3$.

Stable Models

- Given an interpretation $M \subseteq \mathrm{Hb}(P)$, the Gelfond-Lifschitz reduct

$$P^M = \{\mathrm{head}(r) \leftarrow \mathrm{body}^+(r) \mid r \in P \text{ and } \mathrm{body}^-(r) \cap M = \emptyset\}.$$

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

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

$$P^{M_1} = \{\mathsf{a} \leftarrow\} \text{ and } P^{M_2} = P^{M_3} = \emptyset.$$

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

Supported Models

- Given an interpretation $M \subseteq \mathrm{Hb}(P)$, we define the set of *supporting rules*

$$\mathrm{SR}(P, M) = \{r \in P \mid M \models \mathrm{body}(r)\}.$$

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

Example: A positive program $P = \{\mathsf{a} \leftarrow \mathsf{b};\ \mathsf{b} \leftarrow \mathsf{a}\}$ has two supported models $M_1 = \emptyset$ and $M_2 = \{\mathsf{a}, \mathsf{b}\}$, 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).*

2. GENERAL ASSUMPTIONS ON LOGIC PROGRAMS

Definition: *A* logic program *is a triple* $\langle P, A, V\rangle$ *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 $\langle P, A, V\rangle$, $\mathrm{Hb_a}(P)$ for A, $\mathrm{Hb}(P)$ for the set of atoms appearing in P and A, and $\mathrm{Hb_v}(P)$ for V. The **hidden** part $\mathrm{Hb_h}(P)$ is $\mathrm{Hb}(P) - \mathrm{Hb_v}(P)$.

Unless othwerwise stated $\mathrm{Hb_a}(P) = \emptyset$ and $\mathrm{Hb_v}(P) = \mathrm{Hb}(P)$.

Example: A logic program $P = \{\mathsf{a} \leftarrow \sim\mathsf{a}\}$ with $\mathrm{Hb}(P) = \{\mathsf{a}, \mathsf{b}\}$ and $\mathrm{Hb_v}(P) = \{\mathsf{a}\}$ has two *classical models* $M_1 = \{\mathsf{a}\}$ and $M_2 = \{\mathsf{a}, \mathsf{b}\}$.

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 $\mathrm{Hb}(P)$ 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 $\mathrm{Sem}_{\mathcal{C}}$ that maps a program $P \in \mathcal{C}$ to a set of sets $\mathrm{Sem}_{\mathcal{C}}(P) \subseteq \mathbf{2}^{\mathrm{Hb}(P)}$, i.e., the **set of models** of P.

Example: The class of finite normal programs $\mathcal{P}$ satisfies these criteria but $\mathcal{P}_{\mathrm{odd}} = \{P \in \mathcal{P} \mid P \text{ has an odd number of rules}\}$ 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).

☞ $\mathcal{A} \subset \mathcal{U} \subset \mathcal{B} \subset \mathcal{P}$.

➢ For each class $\mathcal{C} \in \{\mathcal{A}, \mathcal{U}, \mathcal{B}, \mathcal{P}\}$, the semantics is determined by

$$\mathrm{Sem}_{\mathcal{C}}(P) = \mathrm{SM}(P) = \{M \subseteq \mathrm{Hb}(P) \mid M = \mathrm{LM}(P^M)\}.$$

➢ The classes of positive programs $\mathcal{A}^+ \subset \mathcal{U}^+ \subset \mathcal{B}^+ \subset \mathcal{P}^+$ are obtained analogously by denying negative body literals.

Example: Sets of Clauses

➢ In analogy to rules, propositional clauses

$$\mathsf{a}_1 \vee \cdots \vee \mathsf{a}_n \vee \neg\mathsf{b}_1 \vee \cdots \vee \neg\mathsf{b}_m$$

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

$$\mathrm{Sem}_{\mathcal{SC}}(S) = \mathrm{CM}(S) = \{M \subseteq \mathrm{Hb}(S) \mid M \models S\}.$$

☞ $\mathcal{SC}$ 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 $P \equiv_{\mathrm{v}} Q$) $\iff$*

1. *$\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(Q)$ and*
2. *there is a bijective function $f : \mathrm{Sem}_{\mathcal{C}}(P) \to \mathrm{Sem}_{\mathcal{C}'}(Q)$ such that*

$$M \cap \mathrm{Hb}_{\mathrm{v}}(P) = f(M) \cap \mathrm{Hb}_{\mathrm{v}}(Q).$$

holds for every $M \in \mathrm{Sem}_{\mathcal{C}}(P)$.

➢ 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 $\equiv_{\mathrm{v}}$.

Example:

1. The stable models of a normal logic program

$$P = \{\mathsf{a} \leftarrow \sim\mathsf{b};\ \ \mathsf{b} \leftarrow \sim\mathsf{a};\ \ \mathsf{c} \leftarrow \mathsf{a};\ \ \mathsf{c} \leftarrow \sim\mathsf{a}\}$$

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

2. The set of clauses

$$S = \{\mathsf{a} \vee \mathsf{d},\ \neg\mathsf{a} \vee \neg\mathsf{d},\ \mathsf{a} \vee \mathsf{c},\ \neg\mathsf{a} \vee \mathsf{c}\}$$

   has exactly two classical models $N_1 = \{\mathsf{a}, \mathsf{c}\}$ and $N_2 = \{\mathsf{d}, \mathsf{c}\}$, since $\mathrm{Hb}(S) = \{\mathsf{a}, \mathsf{c}, \mathsf{d}\}$.

The atoms b and d, which appear in P and S, respectively, can be hidden by tuning the visibility of atoms: $\mathrm{Hb_v}(P) = \mathrm{Hb_v}(S) = \{\mathsf{a}, \mathsf{c}\}$.

☞ $P \equiv_{\mathrm{v}} S$ holds.

Alternative notions of equivalence

Let us compare $\equiv_{\mathrm{v}}$ with the following relations:

Definition: *Programs $P \in \mathcal{P}$ and $Q \in \mathcal{P}$ are **(weakly) equivalent*** $\iff$ $\mathrm{SM}(P) = \mathrm{SM}(Q)$.

Definition: *Programs $P \in \mathcal{P}$ and $Q \in \mathcal{P}$ are **strongly equivalent*** $\iff$ $\mathrm{SM}(P \cup R) = \mathrm{SM}(Q \cup R)$ *for all* $R \in \mathcal{P}$.

Definition: *Programs $P \in \mathcal{C}$ and $Q \in \mathcal{C}'$ are **weakly visibly equivalent** (denoted by $P \equiv_{\mathrm{w}} Q$)* $\iff$

1. $\mathrm{Hb_v}(P) = \mathrm{Hb_v}(Q)$ *and*
2. $\{M \cap \mathrm{Hb_v}(P) \mid M \in \mathrm{Sem}_{\mathcal{C}}(P)\} = \{N \cap \mathrm{Hb_v}(Q) \mid N \in \mathrm{Sem}_{\mathcal{C}'}(Q)\}$.

4. PROPERTIES OF TRANSLATION FUNCTIONS

Definition: *A translation function $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ is **polynomial (P)*** $\iff$ *for all $P \in \mathcal{C}$, the time required to compute $\mathrm{Tr}(P)$ is polynomial in $||P||$, i.e. the number of symbols needed to represent P.*

Definition: *A translation function $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ is **faithful (F)*** $\iff$ *for all $P \in \mathcal{C}$, $P \equiv_{\mathrm{v}} \mathrm{Tr}(P)$.*

Example: Consider a hypothetical translation function

$$\mathrm{Tr}_{\mathrm{DOUBLE}}(P) = P \cup \{\mathsf{a} \leftarrow \sim\mathsf{b};\ \ \mathsf{b} \leftarrow \sim\mathsf{a}\},$$

where $\mathsf{a} \notin \mathrm{Hb}(P)$ and $\mathsf{b} \notin \mathrm{Hb}(P)$ are two new atoms, and $\mathrm{Hb_v}(\mathrm{Tr}_{\mathrm{DOUBLE}}(P)) = \mathrm{Hb_v}(P)$ by definition.

☞ $\mathrm{Tr}_{\mathrm{DOUBLE}}$ is linear (and thus polynomial) but not faithful.

Properties of Translation Functions (continued)

Module conditions for two programs $P \in \mathcal{C}$ and $Q \in \mathcal{C}$ are:

M1. $P \cap Q = \emptyset$	M2. $\mathrm{Hb_a}(P) \cap \mathrm{Hb_a}(Q) = \emptyset$
M3. $\mathrm{Hb_h}(P) \cap \mathrm{Hb}(Q) = \emptyset$	M4. $\mathrm{Hb}(P) \cap \mathrm{Hb_h}(Q) = \emptyset$

Definition: *A translation function $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ is **modular (M)*** $\iff$ *for all $P \in \mathcal{C}$ and $Q \in \mathcal{C}$ satisfying M1–M4, $\mathrm{Tr}(P \cup Q) = \mathrm{Tr}(P) \cup \mathrm{Tr}(Q)$; and $\mathrm{Tr}(P)$ and $\mathrm{Tr}(Q)$ satisfy M1–M4.*

Definition: *A translation function $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ is **PFM*** $\iff$ *Tr is polynomial, faithful, and modular.*

Proposition: *Any composition of polynomial/faithful/modular translation functions is also polynomial/faithful/modular.*

Classification Method

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

- $\mathcal{C} \leq_{\mathrm{PFM}} \mathcal{C}'$ (there is a PFM translation function $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$), **or**
- $\mathcal{C} \not\leq_{\mathrm{PFM}} \mathcal{C}'$ (such a translation function does not exist).

These relations induce further relations for classes of logic programs:

Notation	Definition	Relation
$\mathcal{C} <_{\mathrm{PFM}} \mathcal{C}'$	$\mathcal{C} \leq_{\mathrm{PFM}} \mathcal{C}'$ and $\mathcal{C}' \not\leq_{\mathrm{PFM}} \mathcal{C}$	*strictly less*
$\mathcal{C} =_{\mathrm{PFM}} \mathcal{C}'$	$\mathcal{C} \leq_{\mathrm{PFM}} \mathcal{C}'$ and $\mathcal{C}' \leq_{\mathrm{PFM}} \mathcal{C}$	*equal*
$\mathcal{C} \neq_{\mathrm{PFM}} \mathcal{C}'$	$\mathcal{C} \not\leq_{\mathrm{PFM}} \mathcal{C}'$ and $\mathcal{C}' \not\leq_{\mathrm{PFM}} \mathcal{C}$	*incomparable*

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

5. EXPRESSIVE POWER ANALYSIS

Proposition: *The inclusions* $\mathcal{A}^+ \subset \mathcal{U}^+ \subset \mathcal{B}^+ \subset \mathcal{P}^+$ *imply* $\mathcal{A}^+ \leq_{\mathrm{PFM}} \mathcal{U}^+ \leq_{\mathrm{PFM}} \mathcal{B}^+ \leq_{\mathrm{PFM}} \mathcal{P}^+$.

Theorem: $\mathcal{U}^+ \not\leq_{\mathrm{FM}} \mathcal{A}^+$

Proof. Suppose that $\mathrm{Tr} : \mathcal{U}^+ \to \mathcal{A}^+$ is faithful and modular.

Programs $P = \{\mathsf{a} \leftarrow \mathsf{b}\}$ and $Q = \{\mathsf{b} \leftarrow\}$ satisfy M1–M4.

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

Now $\mathsf{a} \in \mathrm{LM}(P \cup Q)$ $\implies$ $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}(P) \cup \mathrm{Tr}(Q))$
$\implies$ $\mathsf{a} \leftarrow$ belongs to $\mathrm{Tr}(P)$ or to $\mathrm{Tr}(Q)$
$\implies$ $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}(P))$ or $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}(Q))$
$\implies$ $\mathsf{a} \in \mathrm{LM}(P)$ or $\mathsf{a} \in \mathrm{LM}(Q)$,
a contradiction.

Expressiveness of Positive Programs (Continued)

- However, a faithful and **non-modular** translation function from $\mathcal{U}^+$ to $\mathcal{A}^+$ is still possible:

$$\mathrm{Tr}_{\mathrm{LM}}(P) = \{\mathsf{a} \leftarrow \mid \mathsf{a} \in \mathrm{LM}(P)\}.$$

- For the programs P and Q in the preceding counter-example: $\mathrm{Tr}_{\mathrm{LM}}(P) = \emptyset$, $\mathrm{Tr}_{\mathrm{LM}}(Q) = \{\mathsf{b} \leftarrow\}$ and $\mathrm{Tr}_{\mathrm{LM}}(P \cup Q) = \{\mathsf{a} \leftarrow;\ \mathsf{b} \leftarrow\} \neq \mathrm{Tr}_{\mathrm{LM}}(P) \cup \mathrm{Tr}_{\mathrm{LM}}(Q)$.
- Moreover, it can be established that $\mathcal{B}^+ \not\leq_{\mathrm{FM}} \mathcal{U}^+$ and $\mathcal{P}^+ \leq_{\mathrm{PFM}} \mathcal{B}^+$.
- The resulting expressive power hierarchy for positive programs:

$$\mathcal{A}^+ <_{\mathrm{PFM}} \mathcal{U}^+ <_{\mathrm{PFM}} \mathcal{B}^+ =_{\mathrm{PFM}} \mathcal{P}^+.$$

Summary of our Results for Normal Programs

- $\mathcal{A} \subset \mathcal{U} \subset \mathcal{B} \subset \mathcal{P} \implies \mathcal{A} \leq_{\mathrm{PFM}} \mathcal{U} \leq_{\mathrm{PFM}} \mathcal{B} \leq_{\mathrm{PFM}} \mathcal{P}$.
- Non-binary rules can be translated away: $\mathcal{P} \leq_{\mathrm{PFM}} \mathcal{B}$.
- Binary and unary rules cannot be translated away in a **faithful and modular** way: $\mathcal{B} \not\leq_{\mathrm{FM}} \mathcal{U}$ and $\mathcal{U} \not\leq_{\mathrm{FM}} \mathcal{A}$.
- It is straightforward to encode propositional satisfiability problems as (atomic) normal programs: $\mathcal{SC} \leq_{\mathrm{PFM}} \mathcal{A}$.
- Due to non-monotonicity $\mathcal{A} \not\leq_{\mathrm{FM}} \mathcal{SC}$.
- The resulting hierarchy of the five classes under consideration:

$$\mathcal{SC} <_{\mathrm{PFM}} \mathcal{A} <_{\mathrm{PFM}} \mathcal{U} <_{\mathrm{PFM}} \mathcal{B} =_{\mathrm{PFM}} \mathcal{P}.$$

6. YET ANOTHER CHARACTERIZATION OF STABILITY

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

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

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

☞ 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* $\mathsf{a} \in M$, $\#\mathsf{a}$ *is defined as for the members of* $\mathrm{lfp}(\mathrm{T}_{P^M})$.
2. *For* $r \in \mathrm{SR}(P, M)$, $\#r = \max[\{1\} \cup \{\#\mathsf{b} + 1 \mid \mathsf{b} \in \mathrm{body}^+(r)\}]$.

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

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

☞ 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 $\mathrm{Tr}_{\mathrm{AT}}(P)$.

➢ 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 $\mathsf{a} \in M$ and rules $r \in \mathrm{SR}(P, M)$.

➢ The result is to be a polynomial and faithful translation function such that $||\mathrm{Tr}(P)||$ is of order $||P|| \times \log_2 |\mathrm{Hb}(P)|$.

Capturing Supported Models: $\mathrm{Tr}_{\mathrm{SUPP}}(P)$

➢ The complementary atom $\overline{\mathsf{a}}$ is defined for each $\mathsf{a} \in \mathrm{Hb}(P)$:

$$\overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a}.$$

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

$$\mathsf{bt}(r) \leftarrow {\sim}\overline{\mathrm{body}^+(r)}, {\sim}\mathrm{body}^-(r),$$
$$\overline{\mathsf{bt}(r)} \leftarrow {\sim}\mathsf{bt}(r), \text{ and}$$
$$\mathrm{head}(r) \leftarrow {\sim}\overline{\mathsf{bt}(r)}$$

where $\mathsf{bt}(r)$ 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: $\mathrm{Tr}_{\mathrm{CTR}}(P)$

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

Checking Maximality: $\mathrm{Tr}_{\mathrm{MAX}}(P)$

- The value of $\mathsf{ctr}(r)$ is supposed to be $\#r$ in binary.
- If $\mathrm{body}^+(r) \neq \emptyset$, we need for each $\mathsf{b} \in \mathrm{body}^+(r)$ subprograms

  $\mathrm{LT}_{\nabla P}(\mathsf{ctr}(r), \mathsf{nxt(b)}, \overline{\mathsf{bt}(r)})$ and $\mathrm{EQ}_{\nabla P}(\mathsf{ctr}(r), \mathsf{nxt(b)}, \overline{\mathsf{bt}(r)})$

  plus the following rules:

$$\mathsf{x} \leftarrow \sim\mathsf{x}, \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt(b)})_1};$$

$$\mathsf{max}(r) \leftarrow \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{eq}(\mathsf{ctr(r)}, \mathsf{nxt(b)})}; \quad \text{and}$$

$$\mathsf{x} \leftarrow \sim\mathsf{x}, \sim\overline{\mathsf{bt}(r)}, \sim\mathsf{max}(r).$$

- The case that $\mathrm{body}^+(r) = \emptyset$ is handled by $\mathrm{Tr}_{\mathrm{CTR}}(P)$.

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

- The value of $\mathsf{ctr(a)}$ is supposed to be $\#\mathsf{a}$ in binary.
- For each rule r and $\mathsf{a} = \mathrm{head}(r)$, we need the subprograms

  $\mathrm{LT}_{\nabla P}(\mathsf{ctr}(r), \mathsf{ctr(a)}, \overline{\mathsf{bt}(r)})$ and $\mathrm{EQ}_{\nabla P}(\mathsf{ctr}(r), \mathsf{ctr(a)}, \overline{\mathsf{bt}(r)})$

  in addition to the following rules:

$$\mathsf{y} \leftarrow \sim\mathsf{y}, \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{ctr(a)})_1} \text{ and}$$

$$\mathsf{min(a)} \leftarrow \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr(a)})}.$$

- For each $\mathsf{a} \in \mathrm{Hb}(P)$, we introduce the rule $\mathsf{y} \leftarrow \sim\mathsf{y}, \sim\overline{\mathsf{a}}, \sim\mathsf{min(a)}$.

☞ The translation function $\mathrm{Tr}_{\mathrm{AT}}$ defined by $\mathrm{Tr}_{\mathrm{AT}}(P) = \mathrm{Tr}_{\mathrm{SUPP}}(P) \cup \mathrm{Tr}_{\mathrm{CTR}}(P) \cup \mathrm{Tr}_{\mathrm{CTR}}(P) \cup \mathrm{Tr}_{\mathrm{MIN}}(P)$ is both sub-quadratic (thus also polynomial) and faithful.

Non-Modular Translation Functions (Continued)

- The next objective is to embed $\mathcal{A}$ into $\mathcal{SC}$.

  Definition: *For an atomic normal program $P \in \mathcal{A}$ and an atom $\mathsf{a} \in \mathrm{Hb}(P)$, let $\mathrm{Def}_P(\mathsf{a}) = \{r \in P \mid \mathrm{head}(r) = \mathsf{a}\}$,*

$$\begin{aligned}\mathrm{Tr}_{\mathrm{CL}}(\mathsf{a}, P) = {} & \{\mathsf{a} \vee \neg\mathsf{bt}(r) \mid \mathsf{a} \in \mathrm{Hb}(P) \text{ and } r \in \mathrm{Def}_P(\mathsf{a})\} \cup \\ & \{\neg\mathsf{a} \vee \bigvee\{\mathsf{bt}(r) \mid r \in \mathrm{Def}_P(\mathsf{a})\} \mid \mathsf{a} \in \mathrm{Hb}(P)\} \cup \\ & \{\mathsf{bt}(r) \vee \bigvee \mathrm{body}^-(r) \mid r \in \mathrm{Def}_P(\mathsf{a})\} \cup \\ & \{\neg\mathsf{bt}(r) \vee \neg\mathsf{c} \mid r \in \mathrm{Def}_P(\mathsf{a}) \text{ and } \mathsf{c} \in \mathrm{body}^-(r)\},\end{aligned}$$

  and $\mathrm{Tr}_{\mathrm{CL}}(P) = \bigcup_{\mathsf{a} \in \mathrm{Hb}(P)} \mathrm{Tr}_{\mathrm{CL}}(\mathsf{a}, P)$.

☞ $\mathcal{A} \leq_{\mathrm{PF}} \mathcal{SC}$, $\mathcal{P} \leq_{\mathrm{PF}} \mathcal{SC}$, and $\mathcal{SC} =_{\mathrm{PF}} \mathcal{A} =_{\mathrm{PF}} \mathcal{U} =_{\mathrm{PF}} \mathcal{B} =_{\mathrm{PF}} \mathcal{P}$.

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 $\mathsf{a} \leftarrow \mathsf{b}, \sim\mathsf{c}$ is replaced by

$$\mathsf{a} \leftarrow \sim\mathsf{b}_1, \sim\mathsf{c} \text{ and } \mathsf{a} \leftarrow \sim\mathsf{b}_1, \sim\mathsf{c}$$

if the definition of b consists of $\mathsf{b} \leftarrow \sim\mathsf{b}_1$ and $\mathsf{b} \leftarrow \sim\mathsf{b}_2$.

☞ 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: $D \equiv_{L(D)} \mathrm{Tr}(D)$.
2. Incrementality: $D_1 \cup D_2 \equiv_{L(D_1) \cup L(D_2)} \mathrm{Tr}(D_1) \cup \mathrm{Tr}(D_2)$.
3. Modularity: $D_1 \cup D_2 \equiv_{L(D_1) \cup L(D_2)} D_1 \cup \mathrm{Tr}(D_2)$.

- 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 $\Longrightarrow$ at least quadratic encoding.
- The stable models of P and the classical models of $\mathrm{Tr}_{\mathrm{ED}}(P)$ are not in a bijective relationship.

Example: Let $P = \{\mathsf{a} \leftarrow \mathsf{b}, \mathsf{c};\ \mathsf{b} \leftarrow \mathsf{d};\ \mathsf{c} \leftarrow \mathsf{d};\ \mathsf{d} \leftarrow \sim\mathsf{e};\ \mathsf{d} \leftarrow \mathsf{a}\}$. The atoms b and c in the unique stable model $M = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$ 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.

➢ U. Egly, et al. [AAAI, 2001]: *Computing Stable Models with Quantified Boolean Formulas: Some Experimental Results.*

- In this approach, disjunctive stable models are captured with **quantified Boolean formulas** $\exists \mathsf{p}_1 \ldots \exists \mathsf{p}_n \forall \mathsf{q}_1 \ldots \forall \mathsf{q}_m\, \phi$.
- 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.

➢ F. Lin and Y. Zhao [AAAI, 2002]: *ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers.*

- 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): $\mathcal{SC} <_{\mathrm{PFM}} \mathcal{A} <_{\mathrm{PFM}} \mathcal{U} <_{\mathrm{PFM}} \mathcal{B} =_{\mathrm{PFM}} \mathcal{P}$.
- EPH (PF): $\mathcal{SC} =_{\mathrm{PF}} \mathcal{A} =_{\mathrm{PF}} \mathcal{U} =_{\mathrm{PF}} \mathcal{B} =_{\mathrm{PF}} \mathcal{P}$.
- Distinctive features of the counter-based approach:
  1. bijective relationship of models and
  2. $||\mathrm{Tr}(P)||$ is of order $||P|| \times \log_2 |\mathrm{Hb}(P)|$.
- Transitive closure can be properly captured with classical models.
- Experimental results with the implementations of $\mathrm{Tr}_{\mathrm{AT}}$ and $\mathrm{Tr}_{\mathrm{CL}}$ are promising, but further optimizations should be pursued for in order to really compete with SMODELS.