Helsinki University of Technology Laboratory for Theoretical Computer Science
Research Reports 82
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 82
Espoo 2003 HUT-TCS-A82

TRANSLATABILITY AND INTRANSLATABILITY RESULTS FOR CERTAIN CLASSES OF LOGIC PROGRAMS

Tomi Janhunen

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Helsinki University of Technology Laboratory for Theoretical Computer Science
Research Reports 82
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 82
Espoo 2003 HUT-TCS-A82

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

Teknillinen korkeakoulu
Tietotekniikan osasto
Tietojenkäsittelyteorian laboratorio

Distribution:
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O.Box 5400
FIN-02015 HUT
Tel. +358-0-451 1
Fax. +358-0-451 3369
E-mail: lab@tcs.hut.fi

ISBN 951-22-6832-9
ISSN 1457-7615

Multiprint Oy
Espoo 2003

ABSTRACT: In this report, we compare the expressive powers of classes of logic programs that are obtained by constraining the number n of atoms in the bodies of rules. This gives rise to the classes of binary programs ($n \leq 2$), unary programs ($n \leq 1$) and atomic programs ($n = 0$). The comparison is based on the existence/nonexistence of polynomial, faithful, and modular (PFM) translation functions between the classes. As a result, we obtain a strict ordering on the classes of logic programs under consideration under the least model semantics. Binary programs are shown to be as expressive as unconstrained programs but strictly more expressive than unary programs. In addition, unary programs are strictly more expressive than atomic programs. This setting remains valid even if we consider normal logic programs, in which negative literals may appear in the bodies of rules, under the stable model semantics. We also take propositional satisfiability into consideration and establish that atomic programs are strictly more expressive than sets of clauses under classical semantics. Finally, we study polynomial, faithful, but non-modular alternatives to PFM translation functions that do not exist. We obtain a breakthrough in this respect by showing that an arbitrary normal logic program P can be reduced to set of clauses using a faithful translation function in time proportional to $||P|| \times \log_2 |\mathrm{Hb}(P)|$. As an implication of this result, the classes of logic programs under consideration become equally expressive if measured in terms of polynomial and faithful translation functions.

KEYWORDS: Modularity, Translation function, Expressive power

CONTENTS

1 Introduction **1**

2 Preliminaries **3**
2.1 Normal Logic Programs 3
2.2 Stable Model Semantics 4
2.3 Syntactic Restrictions 5
2.4 Sets of Clauses 6

3 Translation Functions **7**
3.1 General Assumptions about Logic Programs 7
3.2 Visible Equivalence 9
3.3 Requirements for Translation Functions 12
3.4 Classification Method 14

4 Expressive Power Analysis **17**
4.1 Some Properties of Programs Modules 17
4.2 Positive Programs 19
4.3 Normal Programs 22
4.4 Comparison with Propositional Logic 30

5 Yet Another Characterization of Stability **34**

6 Non-modular Translation Functions **37**
6.1 Positive Programs Revisited 37
6.2 Translating Normal Programs into Atomic Ones 38
Representing Binary Counters 39
A Non-Modular Translation Function $\mathrm{Tr}_{\mathrm{AT}}$ 42
Correctness of the Translation Function $\mathrm{Tr}_{\mathrm{AT}}$ 47
6.3 Propositional Logic Revisited 50

7 Related Work **55**

8 Conclusions **60**
8.1 Future Work 62
8.2 Acknowledgments 62

References **64**

9 Appendix: Proofs **68**
9.1 Correctness of SEL_j 68
9.2 Correctness of NXT_j 68
9.3 Correctness of FIX_j 70
9.4 Correctness of LT_j 70
9.5 Correctness of EQ_j 72
9.6 Correctness of $\mathrm{Tr}_{\mathrm{SUPP}}(P)$ 73
9.7 Correctness of $\mathrm{Tr}_{\mathrm{CTR}}(P)$ 76
9.8 Correctness of $\mathrm{Tr}_{\mathrm{MAX}}(P)$ 78
9.9 Correctness of $\mathrm{Tr}_{\mathrm{MIN}}(P)$ 82

9.10 Correctness of $\mathrm{Tr}_{\mathrm{AT}}(P)$ 85

To the Memory of My Father

1 INTRODUCTION

In *logic programming* [44], a simple rule-based language is used for knowledge representation in a declarative way so that queries can be answered using a procedure based on the resolution principle [39]. Clark [7] proposed soon a form of negation, namely *negation as failure* to prove, to enhance the knowledge representation capabilities of logic programs. This is how the class of *general* or *normal* logic programs was first established, but it turned out difficult to incorporate the negation as failure principle into resolution procedures (c.f. SLDNF-resolution in [32]) in a satisfactory way. For example, the order in which rules are considered by the resolution procedure may affect answers that are given to the queries. Such a feature makes logic programming with negation as failure less declarative and dependent on the implementation of the resolution procedure.

Fortunately, Gelfond and Lifschitz came up with a solution to this problem: *stable model semantics* [16] for normal logic programs. In this approach, negative literals in rules are interpreted simultaneously which restores the declarative nature of programming with rules. Moreover, the emphasis is more on computing stable models (or *answer sets* [17]) for a given logic program rather than using a resolution procedure for query answering. This is because a problem at hand is typically formulated as a logic program so that stable models correspond to the solutions of the problem. Due to recent interest in this approach, *answer set programming* is nowadays considered as a self-standing constraint programming paradigm [35, 36].

The success of stable models and answer set programming is much due to efficient implementations, such as SMODELS [41] and DLV [11], which were developed during the last decade. This work involved development of search algorithms for computing stable models. The basic technique is to use a *branch and bound* algorithm [42] and the *well-founded model* [45] as an approximation to bound search space. However, even tighter approximations can be devised given assumptions on stable models to be computed. Such principles have been incorporated to the search algorithm behind SMODELS [41], for instance. One such principle tries to apply rules *contrapositively*: if the head h of a rule $\mathsf{h} \leftarrow \mathsf{a}_1, \ldots, \mathsf{a}_n$ is known/assumed to be false in a stable model M being computed, then one of the atoms $\mathsf{a}_1, \ldots, \mathsf{a}_n$ in the body must also be false in M. In particular, this principle becomes effective when $n = 1$ or when all atoms among $\mathsf{a}_1, \ldots, \mathsf{a}_n$ except a_i are known/assumed to be true in M. Then a_1, or respectively a_i in the latter case, must be false in M, and in this way, our knowledge/assumptions on the stable model M being computed (i.e. the approximation) is refined a bit by the truth value assigned to a_1 or a_i in general. These observations suggest that the use of this principle could be accelerated if the number of atoms in the bodies of rules were reduced somehow. Consequently, we are faced with a fundamental question if such a reduction is possible and feasible in the first place.

To address and answer this question, this report concentrates on analyzing the expressive powers of logic program classes that are obtained by restricting the number n of atoms in the bodies of rules. The analysis builds on three syntactic subclasses, namely *binary programs* ($n \leq 2$), *unary programs* ($n \leq 1$) and *atomic programs* ($n = 0$). The central idea is to develop and

apply a method that is similar to the one used by the author [22, 26] for classifying non-monotonic logics by their expressive power. A basic step in such analysis is to check the existence of a *polynomial, faithful and modular* (PFM) translation function between two classes. If we are able to prove that such a translation function does not exist between certain classes, then the syntactic constraints imposed on the classes are significant and affect expressive power. In particular, if there is no PFM translation function that reduces the number of body atoms, then the reduction is likely to be infeasible in practice as it cannot be done in a systematic (modular) way. In fact, the results established in this report indicate that the number of body atoms can be reduced down to two, but going below that is impossible in a faithful and modular way. In other words, arbitrary programs can be translated into binary ones in this way, but binary/unary programs cannot be translated into unary/atomic ones. In order to understand these intranslatability results better, we will also address non-modular alternatives in those cases where modular translations turn out to be impossible.

On the other hand, many problems that have been solved using answer set programming methodology have also formulations as classical satisfiability problems. However, such formulations tend to be more difficult and less concise. For example, formulating an AI planning problem is much easier as a normal logic program [8] than as a set of clauses [28]. This suggests that there is a real difference in expressive power that could be formally established using the methods devised in this report. For this reason, we take propositional satisfiability (i.e., sets of clauses under classical models) into consideration in order to compare it with the classes of logic programs distinguished in this report. It turns out that propositional satisfiability forms a class that is strictly less expressive than the classes of normal logic programs addressed in this report.

The rest of this report is organized as follows. In Section 2, we review the syntax and semantics of the formalisms addressed in this report: normal logic programs under stable model semantics and sets of clauses under classical semantics. Moreover, we distinguish syntactic subclasses of normal logic programs which are taken into consideration. To prepare forthcoming expressiveness analysis, we distinguish three fundamental properties of translation functions in Section 3: polynomiality, faithfulness and modularity. Consequently, a method for comparing the expressive powers of classes of logic programs is established. The actual expressiveness analysis takes place in Section 4. The classes of logic programs are ordered on the basis of their expressive powers which gives rise to an expressive power hierarchy for the classes under consideration. These comparisons involve intranslatability results that count on the modularity property. However, this leaves us the possibility of obtaining faithful, polynomial, but non-modular translation functions. Such alternatives are pursued in Section 6. A particular objective in this respect is to translate faithfully arbitrary logic programs into sets of clauses in polynomial time. As a preparatory step in this respect, we present an alternative characterization of stable models in Section 5. Finally, we perform comparisons with related work in Section 7 and the report ends with conclusions in Section 8.

2 PRELIMINARIES

In this section, we review the basic terminology and definitions of logic programs as well as propositional satisfiability. The syntax of normal logic programs is explained in Section 2.1 while Section 2.2 concentrates on their semantics. In Section 2.3 we introduce syntactic restrictions that play the key role in forthcoming expressiveness analysis. The last subsection, namely Subsection 2.4, reviews the details of sets of propositional clauses as well as their classical model-theoretic semantics.

2.1 Normal Logic Programs

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

$$\mathsf{a} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_n, {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m \tag{2.1}$$

where a is an atom, and $\{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$ and $\{\mathsf{c}_1, \ldots, \mathsf{c}_m\}$ form sets of atoms. In this paper, we restrict ourselves to the purely propositional case and consider only programs that consist of propositional atoms[1]. The symbol $\sim$ denotes *default negation* or *negation as failure to prove* [7] which differs in an important way from *classical negation* denoted by $\neg$. We define *default literals* in the standard way using $\sim$ as negation. The informal intuition behind a rule (2.1) within a normal program is that the atom a can be inferred using the rule given that the atoms $\mathsf{b}_1, \ldots, \mathsf{b}_n$ can be inferred using some other rules but none of the atoms $\mathsf{c}_1, \ldots, \mathsf{c}_m$ can be inferred using any other rules. The exact model-theoretic semantics of rules will be given in the next subsection.

Given a rule r of the form (2.1), the atom a forms the *head* of r whereas the positive literals $\mathsf{b}_1, \ldots, \mathsf{b}_n$ and the negative literals ${\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m$ together form the *body* of r. Despite of the notation used in (2.1), we interpret the body as a set of literals, which implies that the order of the literals is not relevant. To enable easy reference to the atoms/literals in the body, we adopt the following notations: $\mathrm{head}(r) = \mathsf{a}$, $\mathrm{body}(r) = \{\mathsf{b}_1, \ldots, \mathsf{b}_n\} \cup \{{\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m\}$, $\mathrm{body}^+(r) = \{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$, and $\mathrm{body}^-(r) = \{\mathsf{c}_1, \ldots, \mathsf{c}_m\}$. We generalize these notations for any normal program P in the obvious way, e.g. $\mathrm{head}(P) = \{\mathrm{head}(r) \mid r \in P\}$, and $\mathrm{body}(P)$, $\mathrm{body}^+(P)$, and $\mathrm{body}^-(P)$ are analogously defined. The *positive part* r^+ of a rule r is defined as $\mathrm{head}(r) \leftarrow \mathrm{body}^+(r)$.

For now, the *Herbrand base* $\mathrm{Hb}(P)$ of a normal logic program P is defined as the set of atoms that appear in the rules of P, although a slightly more general definition will be introduced in Section 3.1. In the sequel, the class of normal programs is denoted by $\mathcal{P}$. To ease the forthcoming semantical definitions, we distinguish *positive (normal) programs* which are normal programs P whose every rule $r \in P$ satisfy $\mathrm{body}^-(r) = \emptyset$. Consequently, the class of positive programs $\mathcal{P}^+$ forms a proper subclass of $\mathcal{P}$.

[1] Programs with variables, constants and function symbols are covered implicitly through Herbrand instantiation. However, the forthcoming expressiveness analysis is based on finite programs and Herbrand instances, which means that function symbols are not fully covered.

2.2 Stable Model Semantics

Normal programs can be given a standard model-theoretic semantics. An *interpretation* $I \subseteq \mathrm{Hb}(P)$ of a normal program P determines which atoms a of $\mathrm{Hb}(P)$ are *true* ($\mathsf{a} \in I$) and which atoms are *false* ($\mathsf{a} \in \mathrm{Hb}(P) - I$). The satisfaction relation $\models$ is given in Definition 2.1 below. In particular, note that negative default literals that may appear in the bodies of rules are given a classical interpretation at this point: $I \models {\sim}\mathsf{a} \iff I \not\models \mathsf{a}$. Moreover, the interpretation of rules is similar to that of classical implications.

Definition 2.1 *Let $I \subseteq \mathrm{Hb}(P)$ be an interpretation of normal program P.*

1. *For an atom $a \in \mathrm{Hb}(P)$, $I \models \mathsf{a} \iff \mathsf{a} \in I$.*
2. *For a negative literal ${\sim}\mathsf{a}$ based on $\mathsf{a} \in \mathrm{Hb}(P)$, $I \models {\sim}\mathsf{a} \iff \mathsf{a} \notin I$.*
3. *For a set of default literals L, $I \models L \iff \forall l \in L : I \models l$.*
4. *For a rule r, $I \models r \iff I \models \mathrm{body}(r)$ implies $I \models \mathrm{head}(r)$.*
5. *Finally, I is a (classical) model of P ($I \models P$) $\iff \forall r \in P : I \models r$.*

Although the preceding definition gives the classical semantics for arbitrary normal programs P, the ultimate semantics assigned to normal programs will be different as *minimal models* are distinguished.

Definition 2.2 *A model $M \models P$ is a minimal model of P if and only if there is no model $M' \models P$ such that $M' \subset M$.*

In particular, every positive program P is guaranteed to possess a unique minimal model of which equals to the intersection of all models of P [32]. We let $\mathrm{LM}(P)$ stand for this particular model, i.e. the *least model* of P. The least model semantics is inherently *monotonic*: if $P \subseteq P'$ holds for two positive programs, then $\mathrm{LM}(P) \subseteq \mathrm{LM}(P')$. Moreover, the least model $\mathrm{LM}(P)$ can be constructed iteratively as follows (see [32] for a complete treatment). Define an operator T_P on sets of atoms $A \subseteq \mathrm{Hb}(P)$ by setting $\mathrm{T}_P(A) = \{\mathrm{head}(r) \mid r \in P \text{ and } \mathrm{body}^+(r) \subseteq A\}$. The iteration sequence of the operator T_P is defined inductively:

$$\begin{aligned}&\mathrm{T}_P \uparrow 0 = \emptyset,\\ &\mathrm{T}_P \uparrow i = \mathrm{T}_P(\mathrm{T}_P \uparrow i-1) \text{ for } i > 0, \text{ and}\\ &\mathrm{T}_P \uparrow \omega = \textstyle\bigcup_{i<\omega} \mathrm{T}_P \uparrow i.\end{aligned}$$

It follows that $\mathrm{LM}(P) = \mathrm{T}_P \uparrow \omega = \mathrm{lfp}(\mathrm{T}_P)$. This result matches with our intuition on rules: $\mathrm{LM}(P)$ contains only necessarily true atoms of $\mathrm{Hb}(P)$ that can be inferred by using the rules of P recursively. Note that this fixed point is reached with a finite number of iterations if P is finite. In the sequel, we use the construction above to define the *level number* $\mathrm{lev}(\mathsf{a})$ for each atom $\mathsf{a} \in \mathrm{LM}(P)$, i.e. the least natural number i such that $\mathsf{a} \in \mathrm{T}_P \uparrow i$.

Gelfond and Lifschitz [16] propose a way to apply the least model semantics to an arbitrary normal program P. Given an interpretation $M \subseteq \mathrm{Hb}(P)$, i.e. a model candidate, their idea is to reduce P to a positive program

$$P^M = \{r^+ \mid r \in P \text{ and } M \cap \mathrm{body}^-(r) = \emptyset\}. \tag{2.2}$$

In this way, the negative default literals appearing in the bodies of the rules of P are simultaneously interpreted with respect to M. Since the reduct P^M is a positive program, it has a natural semantics determined by the least model $\mathrm{LM}(P^M)$. Equating this model with the model candidate M used to reduce P leads to the following notion of *stability*.

Definition 2.3 ([16]) *An interpretation $M \subseteq \mathrm{Hb}(P)$ of a normal logic program P is a stable model of P $\iff$ $M = \mathrm{LM}(P^M)$.*

Every stable model of P is also a classical model of P in the sense of Definition 2.1, but not necessarily vice versa. In general, a normal logic program need not have a unique stable model (see P_1 in Example 2.4 below) nor a stable models at all (see P_2 in Example 2.4). In contrast to the least models of positive programs, stable models may change in a *non-monotonic* way which implies that conclusions may be retracted under stable model semantics. This is demonstrated below using programs Q_1 and Q_2 in Example 2.4. Normal logic programs are well-suited for a variety of knowledge representation and reasoning tasks. The reader is referred e.g. to [35, 36] for further examples on using normal logic programs in practice.

Example 2.4 Consider the following normal programs.

1. Program $P_1 = \{\mathsf{a} \leftarrow {\sim}\mathsf{b};\ \mathsf{b} \leftarrow {\sim}\mathsf{a}\}$[2] has stable models $\{\mathsf{a}\}$ and $\{\mathsf{b}\}$.
2. Program $P_2 = \{\mathsf{a} \leftarrow {\sim}\mathsf{a}\}$ has no stable models.
3. Programs $Q_1 = \{\mathsf{a} \leftarrow {\sim}\mathsf{b}\}$ and $Q_2 = P_1 \cup \{\mathsf{b} \leftarrow\}$ have unique stable models $M_1 = \{\mathsf{a}\}$ and $M_2 = \{\mathsf{b}\}$, respectively, but $M_1 \not\subseteq M_2$.

2.3 Syntactic Restrictions

As discussed in the introduction, one of the goals of this paper is to examine the possibilities of translating away positive body literals from rules under stable model semantics. To prepare forthcoming analysis, let us distinguish normal/positive programs that are obtained by restricting the number of positive body literals, i.e. $|\mathrm{body}^+(r)|$, allowed in a rule r.

Definition 2.5 *A rule r of a normal program is called **atomic**, **unary** or **binary**, if $|\mathrm{body}^+(r)| = 0$, $|\mathrm{body}^+(r)| \leq 1$, or $|\mathrm{body}^+(r)| \leq 2$, respectively.*

Moreover, we say that a rule r is *strictly* unary if $|\mathrm{body}^+(r)| = 1$, i.e. it is unary and not atomic. *Strictly* binary rules are defined analogously, i.e. $|\mathrm{body}^+(r)| = 2$. We extend these conditions to cover logic programs in the obvious way: a logic program P satisfies any of these conditions given that every rule of P satisfies the condition. For instance, an *atomic normal program* P contains only rules of the form $\mathsf{a} \leftarrow {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m$. The conditions set in Definition 2.5 imply that atomic programs are unary ones and unary programs are binary ones. Consequently, the respective classes of normal programs (denoted by $\mathcal{A}, \mathcal{U}$, and $\mathcal{B}$, respectively) are ordered by inclusion as

[2] We use semicolons ";" to separate rules in logic programs.

follows: $\mathcal{A} \subset \mathcal{U} \subset \mathcal{B} \subset \mathcal{P}$. In the last class, there are also *non-binary* normal programs that have at least one rule r with $|\mathrm{body}^+(r)| > 2$.

The same syntactic restrictions can be applied within the class of positive programs $\mathcal{P}^+$ distinguished in Section 2.2. For instance, any unary positive program consists of rules of the very simple forms $\mathsf{a} \leftarrow$ and $\mathsf{b} \leftarrow \mathsf{c}$. Let us introduce superscripted symbols $\mathcal{A}^+$, $\mathcal{U}^+$, $\mathcal{B}^+$ to denote the respective subclasses of $\mathcal{P}^+$, which are ordered analogously $\mathcal{A}^+ \subset \mathcal{U}^+ \subset \mathcal{B}^+ \subset \mathcal{P}^+$. Moreover, for each class $\mathcal{C}$ among $\mathcal{A}$, $\mathcal{U}$, $\mathcal{B}$, and $\mathcal{P}$, it holds that $\mathcal{C}^+ \subset \mathcal{C}$. Let us also note that the conditions of Definition 2.5 have been carefully designed to be compatible with Gelfond-Lifschitz reduction. If P is a normal program belonging to $\mathcal{C}$ and $M \subseteq \mathrm{Hb}(P)$, then P^M is a member of the respective class $\mathcal{C}^+$. Finally, we emphasize that the semantics of the logic programs in the classes introduced so far is determined by the stable/least model semantics.

2.4 Sets of Clauses

We define *classical literals* in the standard way using classical negation $\neg$ as the connective. A *clause* C is a finite set of classical literals

$$\{\mathsf{a}_1, \ldots, \mathsf{a}_n, \neg\mathsf{b}_1, \ldots, \neg\mathsf{b}_m\} \tag{2.3}$$

representing a disjunction of its constituents. A set of clauses S represents a conjunction of the clauses C contained in S. We let $\mathrm{Hb}(S)$ denote the Herbrand base of a set of clauses S so that interpretations I for S can be defined as subsets of $\mathrm{Hb}(S)$ in analogy to Section 2.2. To enable comparisons with respect to Definition 2.1, we give below the classical model-theoretic semantics of sets of clauses.

Definition 2.6 *Let $I \subseteq \mathrm{Hb}(S)$ be an interpretation of a set of clauses S.*

1. *For an atom $\mathsf{a} \in \mathrm{Hb}(S)$, $I \models \mathsf{a} \iff \mathsf{a} \in I$.*
2. *For a negative literal $\neg\mathsf{a}$ based on $\mathsf{a} \in \mathrm{Hb}(S)$, $I \models \neg\mathsf{a} \iff I \not\models \mathsf{a}$.*
3. *For a clause $C \in S$, $I \models C \iff \exists l \in C : I \models l$.*
4. *Finally, I is a model of S (denoted by $I \models S$) $\iff \forall C \in S : I \models C$.*

Similarly to logic programs, a set of clauses S gives rise to a set of models, but the essential difference is that all classical models are taken into account. A set of clauses S is *satisfiable* if it has at least one model, and *unsatisfiable* otherwise. Let us yet point out that sets of clauses correspond to classical propositional theories in the conjunctive normal form (CNF).

3 TRANSLATION FUNCTIONS

The author has analyzed the expressive powers of *non-monotonic logics* in a systematic fashion [22, 26, 23] which extends previous work by Imielinski [20] and Gottlob [19]. The comparison is based on the existence/non-existence of *polynomial*, *faithful* and *modular* (PFM) translation functions between non-monotonic logics under consideration. As a result of several pairwise comparisons of non-monotonic logics, the expressive power hierarchy (EPH) of non-monotonic logics [26] was gradually established.

In this report, we propose an analogous framework to compare the expressive powers of classes $\mathcal{C}$ of logic programs. However, the framework will be somewhat different from the one used by the author in [24], as the nature of logic programs has to be taken into account. We proceed as follows. As preliminary issues, we distinguish general properties of logic programs in Section 3.1 and propose a notion of equivalence, namely *visible equivalence*, in Section 3.2. This is to prepare the introduction of PFM translation functions between classes of logic programs in Section 3.3. Section 3 ends with a presentation of the resulting classification method in Section 3.4.

3.1 General Assumptions about Logic Programs

At this level of abstraction, logic programs P are understood syntactically as sets of expressions built of propositional atoms. This is to cover also other formalisms in addition to normal logic programs introduced in Section 2.1. There we defined the Herbrand base $\mathrm{Hb}(P)$ as the set of atoms that effectively *appear* in P. Basically, we would like to apply the same principle at this level abstraction, but sometimes there is a need to extend $\mathrm{Hb}(P)$ by certain atoms that do not appear in P. This kind of a setting may arise e.g. when a particular logic program P is being optimized. Suppose that an atom $\mathsf{a} \in \mathrm{Hb}(P)$ is recognized useless in the program P and all of its occurrences (and possibly the rules involved) are removed from P. Thus $\mathsf{a} \notin \mathrm{Hb}(P')$ holds for the resulting program P'. Let us mention $P = \{\mathsf{a} \leftarrow \mathsf{a}\}$ and $P' = \emptyset$ as concrete examples of such programs, whose least models coincide. The fact that a is forgotten in this way impedes the comparison of the two programs in a sense (as to be defined in Section 3.2), since the Herbrand bases become different according to our definition.

Before presenting our solution to this problem, let us discuss a further aspect of Herbrand bases, namely the *visibility* of atoms. When logic programs are developed to solve a problem at hand, it is typical that only a certain portion of atoms that appear in programs are relevant for representing the solutions of the problem. In this setting, programs may contain auxiliary atoms that are only locally relevant and do not appear in other programs formulated for the problem. Consequently, the models/interpretations assigned to two programs may differ already on the basis of auxiliary atoms. A way to handle this situation is to *hide* local information in analogy to conventional programming languages. Rather than introducing a hiding mechanism for atoms, we concentrate on specifying visible atoms, as decided by the programmer. To solve the problems covered by the discussion above, we proceed as follows.

Definition 3.1 *Formally, a logic program is a triple* $\langle P, A, V\rangle$ *where*

1. P *is a set of expressions (rules) built of atoms;*
2. A *is a set of* ***additional*** *atoms; and*
3. V *specifies which atoms appearing in* P *and* A *are considered* ***visible****.*

By a slight abuse of notation we use P to refer to the program despite the additional structure assigned by Definition 3.1. The sets A and V are often left implicit to be referred by the notations $\mathrm{Hb}_{\mathrm{a}}(P)$ and $\mathrm{Hb}_{\mathrm{v}}(P)$, respectively. From now on, we use $\mathrm{Hb}(P)$ to denote the set of atoms that appears in P and A above. As a derived notion, we define the *hidden* part of $\mathrm{Hb}(P)$ as $\mathrm{Hb}_{\mathrm{h}}(P) = \mathrm{Hb}(P) - \mathrm{Hb}_{\mathrm{v}}(P)$. To ease the treatment of programs in the sequel, we make some default assumptions regarding the set of atoms $\mathrm{Hb}(P)$ and $\mathrm{Hb}_{\mathrm{v}}(P)$. Unless otherwise stated, we assume that

- $\mathrm{Hb}(P)$ is minimal, i.e. $\mathrm{Hb}_{\mathrm{a}}(P) = \emptyset$ and $\mathrm{Hb}(P)$ contains only the atoms that actually appear in P; and
- $\mathrm{Hb}_{\mathrm{v}}(P)$ is maximal, i.e. all atoms of $\mathrm{Hb}(P)$ are visible.

Example 3.2 Given $P = \{\mathsf{a} \leftarrow \sim\mathsf{b}\}$, the default interpretation is that the set $\mathrm{Hb}(P) = \{\mathsf{a}, \mathsf{b}\}$, $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}(P) = \{\mathsf{a}, \mathsf{b}\}$, and $\mathrm{Hb}_{\mathrm{h}}(P) = \emptyset$. But if we want to make an exception, we have to add explicitly that $\mathrm{Hb}(P) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and $\mathrm{Hb}_{\mathrm{v}}(P) = \{\mathsf{a}, \mathsf{c}\}$, for example. By these declarations, the hidden part $\mathrm{Hb}_{\mathrm{h}}(P)$ is implicitly assigned to $\{\mathsf{b}\}$. More formally, this would mean specifying a logic program $\langle P, \{\mathsf{c}\}, \{\mathsf{a}, \mathsf{c}\}\rangle$ conforming to Definition 3.1.

The unique stable model of P is $M = \{\mathsf{a}\}$, as there are no rules for b. This suggests a simplification $Q = \{\mathsf{a} \leftarrow\}$ of P. To keep track of b, we may define $\mathrm{Hb}_{\mathrm{a}}(Q) = \{\mathsf{b}\}$ so that $\mathrm{Hb}(Q) = \mathrm{Hb}_{\mathrm{v}}(Q) = \{\mathsf{a}, \mathsf{b}\}$ holds by default.

It is important to understand any extensions to Herbrand bases so that they contribute to the length of the program, too. This would not be the case if some (even infinite) set of atoms were declared as $\mathrm{Hb}(P)$ separately, not as a part of the program. If admitted by the syntax, we take unions and intersections of of logic programs component by component.

Definition 3.3 *The union and intersection of logic programs* $\langle P_1, A_1, V_1\rangle$ *and* $\langle P_2, A_2, V_2\rangle$ *are*

$$\langle P_1 \cup P_2, A_1 \cup A_2, V_1 \cup V_2\rangle \text{ and } \langle P_1 \cap P_2, A_1 \cap A_2, V_1 \cap V_2\rangle,$$

respectively.

As a consequences of Definition 3.3, we have $\mathrm{Hb}(P \cup Q) = \mathrm{Hb}(P) \cup \mathrm{Hb}(Q)$, $\mathrm{Hb}_{\mathrm{v}}(P \cup Q) = \mathrm{Hb}_{\mathrm{v}}(P) \cup \mathrm{Hb}_{\mathrm{v}}(Q)$, and $\mathrm{Hb}_{\mathrm{h}}(P \cup Q) = (\mathrm{Hb}_{\mathrm{h}}(P) - \mathrm{Hb}_{\mathrm{v}}(Q)) \cup (\mathrm{Hb}_{\mathrm{h}}(Q) - \mathrm{Hb}_{\mathrm{v}}(P))$ for any programs P and Q. The hidden parts of Herbrand bases may shrink in this way when logic programs are joined together.

Generally speaking, the set $\mathrm{Hb}_{\mathrm{v}}(P)$ can be understood as a *program interface* of P and it gives the basis for comparing the program P with other programs of interest. The atoms in $\mathrm{Hb}_{\mathrm{h}}(P)$ are to be hidden in any such comparisons. Having now settled our concerns regarding the Herbrand bases of programs, we make some general assumptions on classes of logic programs.

Definition 3.4 *Any class $\mathcal{C}$ of logic programs must satisfy the following.*

A1. Every $P \in \mathcal{C}$ is ***finite****.*

A2. $\mathcal{C}$ is ***closed under unions****, i.e. $\forall P \in \mathcal{C}$: $\forall Q \in \mathcal{C}$: $P \cup Q \in \mathcal{C}$.*

A3. $\mathcal{C}$ is ***closed under intersections****, i.e. $\forall P \in \mathcal{C}$: $\forall Q \in \mathcal{C}$: $P \cap Q \in \mathcal{C}$.*

A4. There is a ***semantic operator*** $\mathrm{Sem}_{\mathcal{C}}$ *for the class $\mathcal{C}$ which maps a program $P \in \mathcal{C}$ to a subset of $\mathbf{2}^{\mathrm{Hb}(P)}$ determining the semantics of P.*

These assumptions will be needed when the requirements for translation functions are formulated in Section 3.3. The first assumption reflects the fact that we take logic programs as potential inputs to translator programs. Hence we must be able to encode any program under consideration as a finite string of symbols. The following two assumptions ensure that logic programs within a class can be combined in arbitrary ways. For instance, the class of normal logic programs with an odd number of rules is *not* in accordance with A2 nor A3. By the fourth assumption, the semantics of each program P in a class $\mathcal{C}$ is determined by a set of *total* [3] interpretations $\mathrm{Sem}_{\mathcal{C}}(P)$. Given an interpretation $I \in \mathrm{Sem}_{\mathcal{C}}(P)$, each atom $\mathsf{a} \in \mathrm{Hb}(P)$ is assigned either to true or false in analogy to Definitions 2.1 and 2.6. Hence we assume two-valued semantics for the classes of logic programs in this report.

Note that the finite fragments of the classes of logic programs $\mathcal{C}$ introduced in Section 2.3 satisfy assumptions A1–A4. In the sequel, we identify these classes as their finite fragments. For instance, the union of two finite binary normal programs is also a finite binary normal program. In particular, the semantics assigned by the respective semantic operator is

$$\mathrm{Sem}_{\mathcal{C}}(P) = \mathrm{SM}(P) = \{M \subseteq \mathrm{Hb}(P) \mid M = \mathrm{LM}(P^M)\}. \tag{3.1}$$

Note that $P^M = P$ holds for any *positive program* P and $M \subseteq \mathrm{Hb}(P)$, which implies that $\mathrm{LM}(P)$ is the unique stable model of P. Hence the stable model semantics and the least model semantics coincide for positive programs.

Assumptions A1–A4 on classes of logic programs are so loose that it is also possible to view the class of finite sets of clauses $\mathcal{SC}$ as a class of logic programs. As explained in Section 2.4, the semantical operator for $\mathcal{SC}$ is based solely on classical models:

$$\mathrm{Sem}_{\mathcal{SC}}(S) = \mathrm{CM}(S) = \{M \subseteq \mathrm{Hb}(S) \mid M \models S\}. \tag{3.2}$$

3.2 Visible Equivalence

Having defined the semantics of logic programs on an abstract level, the next issue is to define the conditions on which two representatives P and Q of a given class of logic programs $\mathcal{C}$ can be considered to be equivalent. It is natural that the answer to this question goes back to semantics. A straightforward notion of equivalence is obtained by equating $\mathrm{Sem}_{\mathcal{C}}(P)$ and $\mathrm{Sem}_{\mathcal{C}}(Q)$. This corresponds to the basic notion of equivalence proposed for normal programs

[3] It is quite possible to generalize A4 to cover *partial models* like the well-wounded model [45], but such models are not addressed in this report.

under stable model semantics, but stronger notions have also been proposed. For instance, Lifschitz et al. [29] consider P and Q *strongly equivalent* given that $\mathrm{Sem}_{\mathcal{C}}(P \cup R) = \mathrm{Sem}_{\mathcal{C}}(Q \cup R)$ for all other programs R. Consequently, the strong equivalence of P and Q implies that P and Q can be freely used to substitute each other. Although such a notion seems attractive at the first glance, a drawback is that it is all too restrictive — allowing only very straightforward semantics-preserving modifications to rules of programs.

Another problem with both approaches is that the models in $\mathrm{Sem}_C(P)$ and $\mathrm{Sem}_C(Q)$ have to be identical subsets of $\mathrm{Hb}(P)$ and $\mathrm{Hb}(Q)$, respectively. Therefore, we propose a notion of equivalence which tries to take the interfaces of logic programs properly into account. The idea is that atoms in $\mathrm{Hb}(P) - \mathrm{Hb}_{\mathrm{v}}(P)$ and $\mathrm{Hb}(Q) - \mathrm{Hb}_{\mathrm{v}}(Q)$ are considered as local to P and Q and negligible as far as the equivalence of the programs is concerned.

Definition 3.5 *Logic programs $P \in \mathcal{C}$ and $Q \in \mathcal{C}'$ are **visibly equivalent**, denoted by $P \equiv_{\mathrm{v}} Q$, if and only if $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(Q)$ and there is a bijection $f : \mathrm{Sem}_{\mathcal{C}}(P) \to \mathrm{Sem}_{\mathcal{C}'}(Q)$ such that for every $M \in \mathrm{Sem}_{\mathcal{C}}(P)$,*

$$M \cap \mathrm{Hb}_{\mathrm{v}}(P) = f(M) \cap \mathrm{Hb}_{\mathrm{v}}(Q). \tag{3.3}$$

Note that this notion of equivalence can be applied both within a single class of logic programs, and between different classes, which may be syntactically and/or semantically different. This is a very important aspect, as we intend to study the interrelationships of such classes of programs.

Example 3.6 Consider 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 $\mathrm{Hb}_{\mathrm{v}}(P) = \{\mathsf{a}, \mathsf{c}\}$ and the stable models $M_1 = \{\mathsf{a}, \mathsf{c}\}$ and $M_2 = \{\mathsf{b}, \mathsf{c}\}$. Thus $\mathrm{Hb}_{\mathrm{h}}(P) = \{\mathsf{b}\}$ is hidden when we compare P with a set of clauses

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

possessing exactly two classical models $N_1 = \{\mathsf{a}, \mathsf{c}\}$ and $N_2 = \{\mathsf{d}, \mathsf{c}\}$, as $\mathrm{Hb}(S) = \{\mathsf{a}, \mathsf{c}, \mathsf{d}\}$. We can hide d by setting $\mathrm{Hb}_{\mathrm{v}}(S) = \{\mathsf{a}, \mathsf{c}\}$. Then $P \equiv_{\mathrm{v}} S$ holds, as $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(S)$ and there is a bijection from $f : \mathrm{SM}(P) \to \mathrm{CM}(S)$ that (i) maps M_1 to N_1 so that $M_1 \cap \mathrm{Hb}_{\mathrm{v}}(P) = \{\mathsf{a}, \mathsf{c}\} = N_1 \cap \mathrm{Hb}_{\mathrm{v}}(Q)$, and (ii) maps M_2 to N_2 so that $M_2 \cap \mathrm{Hb}_{\mathrm{v}}(P) = \{\mathsf{c}\} = N_2 \cap \mathrm{Hb}_{\mathrm{v}}(Q)$.

There are also reasonable alternatives to $\equiv_{\mathrm{v}}$.

Definition 3.7 *Logic programs $P \in \mathcal{C}$ and $Q \in \mathcal{C}'$ are **weakly visibly equivalent**, denoted by $P \equiv_{\mathrm{w}} Q$, if and only if $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(Q)$ and*

$$\{M \cap \mathrm{Hb}_{\mathrm{v}}(P) \mid M \in \mathrm{Sem}_{\mathcal{C}}(P)\} = \{N \cap \mathrm{Hb}_{\mathrm{v}}(Q) \mid N \in \mathrm{Sem}_{\mathcal{C}'}(Q)\}. \tag{3.4}$$

Proposition 3.8 *The relations $\equiv_{\mathrm{v}}$ and $\equiv_{\mathrm{w}}$ are equivalence relations among all programs.*

PROOF. The reflexivity of $\equiv_{\mathrm{v}}$ follows essentially by the identity mapping $i : \mathrm{Sem}_{\mathcal{C}}(P) \to \mathrm{Sem}_{\mathcal{C}}(P)$ for any P from any $\mathcal{C}$. The symmetry of $\equiv_{\mathrm{v}}$ is also easily obtained. Given $P \equiv_{\mathrm{v}} Q$ for any programs P and Q from any classes $\mathcal{C}$ and $\mathcal{C}'$, respectively, the existence of an inverse for a bijection $f : \mathrm{Sem}_{\mathcal{C}}(P) \to \mathrm{Sem}_{\mathcal{C}'}(Q)$ is guaranteed. For the transitivity of $\equiv_{\mathrm{v}}$, let f_1 and f_2 be the bijections involved in the assumed relations $P \equiv_{\mathrm{v}} Q$ and $Q \equiv_{\mathrm{v}} R$, respectively, for $P \in \mathcal{C}_1$, $Q \in \mathcal{C}_2$, and $R \in \mathcal{C}_3$. It is clear that $f_1 \circ f_2$ is also a bijection, and we have for all $M \in \mathrm{Sem}_{\mathcal{C}_1}(P)$ that $M \cap \mathrm{Hb}_{\mathrm{v}}(P) = f_1(M) \cap \mathrm{Hb}_{\mathrm{v}}(Q) = f_2(f_1(M)) \cap \mathrm{Hb}_{\mathrm{v}}(R) = (f_1 \circ f_2)(M) \cap \mathrm{Hb}_{\mathrm{v}}(R)$. Moreover, $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(Q)$ and $\mathrm{Hb}_{\mathrm{v}}(Q) = \mathrm{Hb}_{\mathrm{v}}(R)$ imply $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(R)$. Hence $P \equiv_{\mathrm{v}} R$ is the case.

For $\equiv_{\mathrm{w}}$, the proof is immediate, since $P \equiv_{\mathrm{w}} Q$ is obtained as an intersection of two equality/equivalence relations: the one determined by $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(Q)$ and the other determined by (3.4). □

It is worthwhile to do some comparisons. By setting $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}(P)$ and $\mathrm{Hb}_{\mathrm{v}}(Q) = \mathrm{Hb}(Q)$, the weak relation $\equiv_{\mathrm{w}}$ becomes very close to the notion of weak equivalence discussed in the beginning of this section, if interpreted with respect to the class of normal programs under the stable model semantics. The only difference is the implied additional requirement that $\mathrm{Hb}(P) = \mathrm{Hb}(Q)$ if $\equiv_{\mathrm{w}}$ is to hold. By the approach taken in Section 3.1 this requirement becomes of little account: Herbrand bases are always extendible to meet $\mathrm{Hb}(P) = \mathrm{Hb}(Q)$. Actually, we can state the same about $\equiv_{\mathrm{v}}$ using a generalized notion of weak equivalence: $P \equiv Q$ is defined to hold for $P \in \mathcal{C}$ and $Q \in \mathcal{C}' \iff \mathrm{Sem}_{\mathcal{C}}(P) = \mathrm{Sem}_{\mathcal{C}'}(Q)$. It is clear that $\equiv$ is also an equivalence relation among all programs.

Proposition 3.9 *If* $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}(P)$ *equals to* $\mathrm{Hb}_{\mathrm{v}}(Q) = \mathrm{Hb}(Q)$*, then* $P \equiv Q \iff P \equiv_{\mathrm{w}} Q \iff P \equiv_{\mathrm{v}} Q$.

PROOF. Assume that $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}(P)$ equals to $\mathrm{Hb}_{\mathrm{v}}(Q) = \mathrm{Hb}(Q)$ for some $P \in \mathcal{C}$ and $Q \in \mathcal{C}'$. For the first equivalence, it is sufficient to note that (3.4) reduces to $\mathrm{Sem}_{\mathcal{C}}(P) = \mathrm{Sem}_{\mathcal{C}'}(Q)$ given that $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}(P)$ and $\mathrm{Hb}_{\mathrm{v}}(Q) = \mathrm{Hb}(Q)$. In addition, $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(Q)$ holds by our assumptions. To establish $P \equiv Q \iff P \equiv_{\mathrm{v}} Q$, we note that if $\mathrm{Sem}_{\mathcal{C}}(P)$ and $\mathrm{Sem}_{\mathcal{C}'}(Q)$ consist of identical subsets of $\mathrm{Hb}(P) = \mathrm{Hb}(Q)$, the identity mapping i serves as the bijection involved in Definition 3.5. On the other hand, the existence of such a bijection implies $\mathrm{Sem}_{\mathcal{C}}(P) = \mathrm{Sem}_{\mathcal{C}'}(Q)$ by our assumptions. □

Example 3.10 Recall the normal logic program P and the set of clauses S from Example 3.6. Now $P \equiv_{\mathrm{w}} S$ holds, as $\{M \cap \mathrm{Hb}_{\mathrm{v}}(P) \mid M \in \mathrm{SM}(P)\} = \{\{\mathsf{a}, \mathsf{c}\}, \{\mathsf{c}\}\}$ equals to $\{N \cap \mathrm{Hb}_{\mathrm{v}}(S) \mid N \subseteq \mathrm{Hb}(S) \text{ and } N \models S\}$.

The setting demonstrated by Examples 3.6 and 3.10 is an instance of a more general relationship between $\equiv_{\mathrm{v}}$ and $\equiv_{\mathrm{w}}$.

Proposition 3.11 *For any logic programs* P *and* Q, $P \equiv_{\mathrm{v}} Q \implies P \equiv_{\mathrm{w}} Q$.

PROOF. Suppose $P \equiv_{\mathrm{v}} Q$ and $P \not\equiv_{\mathrm{w}} Q$. The former implies $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(Q)$ and the existence of a bijection $f : \mathrm{Sem}_{\mathcal{C}}(P) \to \mathrm{Sem}_{\mathcal{C}'}(Q)$ such that (3.3) is met by every $M \in \mathrm{Sem}_{\mathcal{C}}(P)$. On the other hand $P \not\equiv_{\mathrm{w}} Q$ implies

that there is $M \in \mathrm{Sem}_{\mathcal{C}}(P)$ such that $M \cap \mathrm{Hb}_{\mathrm{v}}(P) \neq N \cap \mathrm{Hb}_{\mathrm{v}}(Q)$ for all $N \in \mathrm{Sem}_{\mathcal{C}'}(Q)$; or the roles of P and Q can be interchanged. But this is a contradiction, as $M \cap \mathrm{Hb}_{\mathrm{v}}(P) = f(M) \cap \mathrm{Hb}_{\mathrm{v}}(Q)$ and $f(M) \in \mathrm{Sem}_{\mathcal{C}'}(Q)$. □

However, the converse does not hold in general, as to be shown below. The distinction between the two relations becomes soon crucial when we formulate the criteria for a semantics-preserving translation function in Section 3.3.

Example 3.12 Let us modify the normal program P from Example 3.6 by setting $\mathrm{Hb}_{\mathrm{v}}(P) = \{\mathsf{c}\}$. Then $\{M \cap \mathrm{Hb}_{\mathrm{v}}(P) \mid M \in \mathrm{SM}(P)\}$ contains only $\{\mathsf{c}\}$ although $\mathrm{SM}(P) = \{\{\mathsf{a}, \mathsf{c}\}, \{\mathsf{b}, \mathsf{c}\}\}$. Given a set of clauses $S' = \{\mathsf{c}\}$ with $\mathrm{Hb}_{\mathrm{v}}(S') = \mathrm{Hb}(S') = \{\mathsf{c}\}$ we have $\mathrm{CM}(S') = \{\{\mathsf{c}\}\}$. Then $P \equiv_{\mathrm{w}} S'$ holds, but $P \not\equiv_{\mathrm{v}} S'$, because a bijection between $\mathrm{SM}(P)$ and $\mathrm{CM}(S')$ is impossible.

3.3 Requirements for Translation Functions

We are now ready to formulate our criteria for a translation function $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ that transforms logic programs P of one class $\mathcal{C}$ into logic programs $\mathrm{Tr}(P)$ of another class $\mathcal{C}'$. In many cases of interest, the latter class is a subclass or a superclass of $\mathcal{C}$, but it makes also sense to perform translations between classes that are incomparable in this respect (such as $\mathcal{P}$ and $\mathcal{SC}$ introduced so far). It is assumed below that both the source class $\mathcal{C}$ and the target class $\mathcal{C}'$ satisfy assumptions A1–A4 made in Definition 3.4.

Definition 3.13 *A translation function* $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ *is* ***polynomial*** *if and only if for all programs* $P \in \mathcal{C}$*, the time required to compute the translation* $\mathrm{Tr}(P) \in \mathcal{C}'$ *is polynomial in* $||P||$.

Here $||P||$ stands for the *length* of the program P in the number of symbols in a string representation of P. Note that the length of the translation $||\mathrm{Tr}(P)||$ is also polynomial in $||P||$ if Tr is polynomial. Thus the polynomiality requirement serves as a rough upper bound how much time and space may be used to compute a translation. In many cases, this bound is by no means tight, and even linear time translation functions can be devised for particular classes of logic programs. Many times this is also highly desirable to allow efficient transformation of knowledge from one representation to another.

Definition 3.14 *A translation function* $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ *is* ***faithful*** *if and only if for all* $P \in \mathcal{C}$*,* $P \equiv_{\mathrm{v}} \mathrm{Tr}(P)$.

Here we emphasize that $P \equiv_{\mathrm{v}} \mathrm{Tr}(P)$ implies $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P))$ by the definition of $\equiv_{\mathrm{v}}$. Thus a faithful translation function Tr may introduce new atoms, which have to remain invisible, or forget old invisible atoms. Moreover, if we insist on polynomiality, then the number of new atoms gets bounded, too. The possibility of introducing new atoms is a crucial option for translation functions to be presented in this report. This is because new atoms serve as shorthands for more complex logical expressions that save space and enable translation functions between certain classes. The existence of a bijection f between $\mathrm{Sem}_{\mathcal{C}}(P)$ and $\mathrm{Sem}_{\mathcal{C}'}(\mathrm{Tr}(P))$ ensures that the semantics

of P is captured by $\mathrm{Tr}(P)$ to a reasonable degree: there is a one-to-one correspondence of models and the models coincide up to $\mathrm{Hb}_{\mathrm{v}}(P)$. Thus both *brave* and *cautious* conclusions with respect to the models/interpretations in $\mathrm{Sem}_{\mathcal{C}}(P)$ is preserved within the language determined by $\mathrm{Hb}_{\mathrm{v}}(P)$. Moreover, it is often desirable in answer set programming that the models in $\mathrm{Sem}_{\mathcal{C}}(P)$ correspond to the solutions of the problem being solved, and a bijective relationship is required in order to preserve the *number* of solutions. This is an important aspect of translation functions that is often neglected in literature. To understand the need for a tight relationship, consider a variant of Definition 3.14 obtained by changing $\equiv_{\mathrm{v}}$ to $\equiv_{\mathrm{w}}$. Let us say that $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ is *weakly faithful* if and only if $P \equiv_{\mathrm{w}} \mathrm{Tr}(P)$ for all $P \in \mathcal{C}$. For this relation, it is enough to fulfill

$$\{M \cap \mathrm{Hb}_{\mathrm{v}}(P) \mid M \in \mathrm{Sem}_{\mathcal{C}}(P)\} = \{N \cap \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P)) \mid N \in \mathrm{Sem}_{\mathcal{C}'}(\mathrm{Tr}(P))\}. \tag{3.5}$$

When (3.5) holds, it is possible to capture models of P in a sense using $\mathrm{Tr}(P)$, but e.g. counting the models in $\mathrm{Sem}_{\mathcal{C}}(P)$ may become infeasible in terms of Tr, as one model $M \in \mathrm{Sem}_{\mathcal{C}}(P)$ may have exponentially many corresponding models $N \in \mathrm{Sem}_{\mathcal{C}'}(\mathrm{Tr}(P))$ such that $M \cap \mathrm{Hb}_{\mathrm{v}}(P) = N \cap \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P)) = N \cap \mathrm{Hb}_{\mathrm{v}}(P)$. As a corollary of Proposition 3.11, we know that a translation function Tr that is faithful in the sense of Definition 3.14 is also weakly faithful. However, the converse does not hold in general for the reasons just explained. The third requirement for translation functions, namely *modularity*, is based on the following notion of *program modules*.

Definition 3.15 *Any two logic programs $P \in \mathcal{C}$ and $Q \in \mathcal{C}$ satisfy* ***module conditions*** *if and only if*

M1. $P \cap Q = \emptyset$,

M2. $\mathrm{Hb}_{\mathrm{a}}(P) \cap \mathrm{Hb}_{\mathrm{a}}(Q) = \emptyset$,

M3. $\mathrm{Hb}_{\mathrm{h}}(P) \cap \mathrm{Hb}(Q) = \emptyset$, *and*

M4. $\mathrm{Hb}(P) \cap \mathrm{Hb}_{\mathrm{h}}(Q) = \emptyset$.

Intuitively, the first two module conditions make P and Q *disjoint* programs in the sense of Definition 3.1. Thus $P \cup Q$, as given in Definition 3.3, can be understood as a disjoint union of P and Q. The last two requirements make sure that P and Q can only share visible atoms.

Definition 3.16 *A translation function $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ is* ***modular*** *if and only if for all $P \in \mathcal{C}$ and $Q \in \mathcal{C}$ satisfying the module conditions M1–M4,*

$$\mathrm{Tr}(P \cup Q) = \mathrm{Tr}(P) \cup \mathrm{Tr}(Q), \tag{3.6}$$

and the translations $\mathrm{Tr}(P)$ and $\mathrm{Tr}(Q)$ satisfy module conditions M1–M4.

The aim of the modularity condition is to enforce the locality of Tr with respect to subprograms P and Q, which satisfy the module conditions M3 and M4 and can thus be viewed as program modules that interact through visible

atoms only. The conditions M1 and M2 imply that such modules cannot share rules nor atoms extending the Herbrand base. By (3.6), the modules P and Q have to be separately translatable and the translation $\mathrm{Tr}(P \cup Q)$ is obtained as the union of the translations of the modules. In addition, a modular translation function is supposed to preserve module conditions M1–M4, i.e. the respective translations $\mathrm{Tr}(P)$ and $\mathrm{Tr}(Q)$ are supposed to remain disjoint and share only visible atoms. Note that a program P with $\mathrm{Hb}_\mathrm{v}(P) = \mathrm{Hb}(P)$ and $\mathrm{Hb}_\mathrm{h}(P) = \emptyset$ can be arbitrarily partitioned into modules P_1 and P_2 such that $\mathrm{Hb}_\mathrm{v}(P_i) = \mathrm{Hb}(P_i)$ for $i \in \{1, 2\}$ and $\mathrm{Hb}_\mathrm{v}(P) = \mathrm{Hb}_\mathrm{v}(P_1) \cup \mathrm{Hb}_\mathrm{v}(P_2) = \mathrm{Hb}(P_1) \cup \mathrm{Hb}(P_2) = \mathrm{Hb}(P)$. This observation implies that such a program P can be translated rule by rule using a modular translation function. The modularity condition becomes less restrictive when programs share rules or involve hidden atoms and (3.6) need not be applicable.

Proposition 3.17 *If $\mathrm{Tr}_1 : \mathcal{C}_1 \to \mathcal{C}_2$ and $\mathrm{Tr}_2 : \mathcal{C}_2 \to \mathcal{C}_3$ are two polynomial, faithful, or modular translation functions, then their composition $\mathrm{Tr}_1 \circ \mathrm{Tr}_2 : \mathcal{C}_1 \to \mathcal{C}_3$ is also polynomial, faithful, or modular, respectively.*

PROOF. Suppose that both Tr_1 and Tr_2 are polynomial. Then there is a polynomial $p_1 : \mathbb{N} \to \mathbb{N}$ and a Turing machine M_1 such that M_1 computes $\mathrm{Tr}_1(P)$ in at most $p_1(||P||)$ steps for any $P \in \mathcal{C}_1$. An analogous statement holds for a polynomial p_2, a machine M_2, Tr_2, and $\mathcal{C}_2$.

It follows that $(\mathrm{Tr}_1 \circ \mathrm{Tr}_2)(P) = \mathrm{Tr}_2(\mathrm{Tr}_1(P))$ is computed by $M_1 M_2$ in at most $p_1(||P||) + p_2(||\mathrm{Tr}_1(P)||)$ steps. Note that $||\mathrm{Tr}_1(P)|| \leq p_1(||P||)$ holds by the properties of Turing machines. Moreover, p_2 is dominated by a monotonically increasing polynomial $p : \mathbb{N} \to \mathbb{N}$.[4] Thus $p_2(||\mathrm{Tr}_1(P)||) \leq p(||\mathrm{Tr}_1(P)||) \leq p(p_1(||P||))$ holds, the number of steps taken by $M_1 M_2$ is bounded by $p_3(||P||) = p_1(||P||) + p(p_1(||P||))$, and $\mathrm{Tr}_1 \circ \mathrm{Tr}_2$ is polynomial.

Let us then assume that Tr_1 and Tr_2 are faithful, i.e. it holds for any $P \in \mathcal{C}_1$ that $P \equiv_\mathrm{v} \mathrm{Tr}_1(P)$ and $\mathrm{Tr}_1(P) \equiv_\mathrm{v} \mathrm{Tr}_2(\mathrm{Tr}_1(P))$. Then the transitivity of $\equiv_\mathrm{v}$ (recall Proposition 3.11) implies $P \equiv_\mathrm{v} \mathrm{Tr}_2(\mathrm{Tr}_1(P))$, i.e. $P \equiv_\mathrm{v} (\mathrm{Tr}_1 \circ \mathrm{Tr}_2)(P)$.

Then consider the case that Tr_1 and Tr_2 are modular and let P and Q be any programs from $\mathcal{C}_1$ such that the module conditions M1–M4 are satisfied. Using the modularity of Tr_1, we obtain $\mathrm{Tr}_1(P \cup Q) = \mathrm{Tr}_1(P) \cup \mathrm{Tr}_1(Q)$ and the translations $\mathrm{Tr}_1(P)$ and $\mathrm{Tr}_2(Q)$ satisfy the module conditions M1–M4. Then we may appeal to the modularity of Tr_2, to establish $\mathrm{Tr}_2(\mathrm{Tr}_1(P \cup Q)) = \mathrm{Tr}_2(\mathrm{Tr}_1(P) \cup \mathrm{Tr}_1(Q)) = \mathrm{Tr}_2(\mathrm{Tr}_1(P)) \cup \mathrm{Tr}_2(\mathrm{Tr}_1(Q))$ and the translations $\mathrm{Tr}_2(\mathrm{Tr}_1(P))$ and $\mathrm{Tr}_2(\mathrm{Tr}_1(Q))$ satisfy the module conditions M1–M4. It follows that $(\mathrm{Tr}_1 \circ \mathrm{Tr}_2)(P \cup Q) = (\mathrm{Tr}_1 \circ \mathrm{Tr}_2)(P) \cup (\mathrm{Tr}_1 \circ \mathrm{Tr}_2)(Q)$ and the translations $(\mathrm{Tr}_1 \circ \mathrm{Tr}_2)(P)$ and $(\mathrm{Tr}_1 \circ \mathrm{Tr}_2)(Q)$ satisfy the module conditions M1–M4 — indicating that the composition $\mathrm{Tr}_1 \circ \mathrm{Tr}_2$ is modular. □

3.4 Classification Method

The criteria collected in Definitions 3.13–3.16 lead to a method for comparing classes of logic programs on the basis of their expressive power. We say

[4] E.g., given any polynomial $p_2(x) = a_n x^n + \ldots + a_0 x^0$, the polynomial $p(x) = |a_n| x^n + \ldots + |a_0| x^0$ is monotonically increasing and $p_2(x) \leq p(x)$ holds for all $x \in \mathbb{N}$.

Relation	Definition	Explanation
$\mathcal{C} <_{\mathrm{PFM}} \mathcal{C}'$	$\mathcal{C} \leq_{\mathrm{PFM}} \mathcal{C}'$ and $\mathcal{C}' \not\leq_{\mathrm{PFM}} \mathcal{C}$	$\mathcal{C}$ is *strictly less expressive* than $\mathcal{C}'$
$\mathcal{C} =_{\mathrm{PFM}} \mathcal{C}'$	$\mathcal{C} \leq_{\mathrm{PFM}} \mathcal{C}'$ and $\mathcal{C}' \leq_{\mathrm{PFM}} \mathcal{C}$	$\mathcal{C}$ and $\mathcal{C}'$ are *equally expressive*
$\mathcal{C} \neq_{\mathrm{PFM}} \mathcal{C}'$	$\mathcal{C} \not\leq_{\mathrm{PFM}} \mathcal{C}'$ and $\mathcal{C}' \not\leq_{\mathrm{PFM}} \mathcal{C}$	$\mathcal{C}$ and $\mathcal{C}'$ are *incomparable*

Table 1: Relations used by the Classification Method

that a translation function $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ is PFM if and only if it is polynomial, faithful, and modular simultaneously. If there exists such a translation function Tr, we write $\mathcal{C} \leq_{\mathrm{PFM}} \mathcal{C}'$ to denote that the class $\mathcal{C}'$ *is at least as expressive as* the class $\mathcal{C}$. This is simply because the essentials of any program $P \in \mathcal{C}$ can be captured using the translation $\mathrm{Tr}(P) \in \mathcal{C}'$. In certain cases, we are able to construct a counter-example which shows that a PFM translation function is impossible, denoted by $\mathcal{C} \not\leq_{\mathrm{PFM}} \mathcal{C}'$. The base relations $\leq_{\mathrm{PFM}}$ and $\not\leq_{\mathrm{PFM}}$ among classes of logic programs form the cornerstones of the classification method – giving rise to relations given in Table 1.

It is sometimes convenient to introduce variants of the relations $\leq_{\mathrm{PFM}}$ and $\not\leq_{\mathrm{PFM}}$ which are obtained by dropping some of the three requirements and the corresponding letter(s) in the notation. For instance, if we establish $\mathcal{C} \not\leq_{\mathrm{FM}} \mathcal{C}'$ for certain classes $\mathcal{C}$ and $\mathcal{C}'$ of logic programs, then $\mathcal{C} \not\leq_{\mathrm{PFM}} \mathcal{C}'$ follows immediately. Also, *non-modular* translation functions will be addressed in this report and the resulting relationships involve $\leq_{\mathrm{PF}}$ rather than $\leq_{\mathrm{PFM}}$. In certain cases, it is easy to establish relationships regarding $\leq_{\mathrm{PFM}}$. By the following, we address a frequently appearing case where the syntax is generalized while the semantics is kept compatible with the original one.

Proposition 3.18 *If $\mathcal{C}$ and $\mathcal{C}'$ are two classes of logic programs such that $\mathcal{C} \subseteq \mathcal{C}'$ and $\mathrm{Sem}_{\mathcal{C}}(P) = \mathrm{Sem}_{\mathcal{C}'}(P)$ for all $P \in \mathcal{C}$, then $\mathcal{C} \leq_{\mathrm{PFM}} \mathcal{C}'$.*

PROOF. Let $\mathrm{Tr}_{\mathrm{ID}}$ be the identity translation function for which $\mathrm{Tr}_{\mathrm{ID}}(P) = P$ for every $P \in \mathcal{C}$. It is clear that $\mathrm{Tr}_{\mathrm{ID}}$ is polynomial. For the faithfulness of $\mathrm{Tr}_{\mathrm{ID}}$, we note that $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$ and the identity mapping $i : \mathrm{Sem}_{\mathcal{C}}(P) \to \mathrm{Sem}_{\mathcal{C}'}(P)$ is the bijection insisted by faithfulness, as $\mathrm{Sem}_{\mathcal{C}}(P) = \mathrm{Sem}_{\mathcal{C}'}(P)$. To see the modularity of $\mathrm{Tr}_{\mathrm{ID}}$, let $P \in \mathcal{C}$ and $Q \in \mathcal{C}$ such that module conditions M1–M4 from Definition 3.15 are satisfied. Now $\mathrm{Tr}_{\mathrm{ID}}(P \cup Q) = P \cup Q = \mathrm{Tr}_{\mathrm{ID}}(P) \cup \mathrm{Tr}_{\mathrm{ID}}(Q)$. The translations $\mathrm{Tr}_{\mathrm{ID}}(P) = P$ and $\mathrm{Tr}_{\mathrm{ID}}(Q) = Q$ satisfy M1 and M2, as P and Q do by assumption. Moreover, $\mathrm{Tr}_{\mathrm{ID}}$ does not affect the visibility of atoms so that $\mathrm{Tr}_{\mathrm{ID}}(P) = P$ and $\mathrm{Tr}_{\mathrm{ID}}(Q) = Q$ meet the conditions M3 and M4, too. □

In many cases, we manage to construct faithful translation functions that only add new hidden atoms to programs being translated. The following theorem shows how such translation functions are proved faithful in general.

Theorem 3.19 *Let $\mathrm{Tr} : \mathcal{C} \to \mathcal{C}'$ be a translation function with the following properties for every $P \in \mathcal{C}$:*

1. $\mathrm{Hb}(P) \subseteq \mathrm{Hb}(\mathrm{Tr}(P))$ *and* $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$*;*

2. *there is an extension function* $\mathrm{Ext} : \mathrm{Sem}_{\mathcal{C}}(P) \to \mathrm{Sem}_{\mathcal{C}'}(\mathrm{Tr}(P))$ *such that* $\forall M \in \mathrm{Sem}_{\mathcal{C}}(P) : M = \mathrm{Ext}(M) \cap \mathrm{Hb}(P)$*; and*

3. *if* $N \in \mathrm{Sem}_{\mathcal{C}'}(\mathrm{Tr}(P))$*, then* $M = N \cap \mathrm{Hb}(P) \in \mathrm{Sem}_{\mathcal{C}}(P)$ *such that* $N = \mathrm{Ext}(M)$*.*

Then Tr *is faithful.*

PROOF. Let P any program from the class $\mathcal{C}$. By the second assumption on Tr, there is a function Ext that maps a model $M \in \mathrm{Sem}_{\mathcal{C}}(P)$ to another $N = \mathrm{Ext}(M)$ in $\mathrm{Sem}_{\mathcal{C}'}(\mathrm{Tr}(P))$. It is established in the sequel that Ext is a bijection that meets the requirements set up in Definition 3.5.

Let us assume that Ext is not injective, i.e. there are two models M_1 and M_2 in $\mathrm{Sem}_{\mathcal{C}}(P)$ such that $M_1 \neq M_2$ and $\mathrm{Ext}(M_1) = \mathrm{Ext}(M_2)$. Thus also $\mathrm{Ext}(M_1) \cap \mathrm{Hb}(P) = \mathrm{Ext}(M_2) \cap \mathrm{Hb}(P)$ holds and $M_1 = M_2$ by the properties of Ext, a contradiction. Hence Ext is necessarily injective. By the third requirement on Tr, we have a projection function Proj that maps $N \in \mathrm{Sem}_{\mathcal{C}'}(\mathrm{Tr}(P))$ to $M = N \cap \mathrm{Hb}(P) \in \mathrm{Sem}_{\mathcal{C}}(P)$. Let us assume that Proj is not injective. Then there are two models $N_1 \neq N_2$ in $\mathrm{Sem}_{\mathcal{C}'}(\mathrm{Tr}(P))$ such that $\mathrm{Proj}(N_1) = \mathrm{Proj}(N_2)$, i.e. $N_1 \cap \mathrm{Hb}(P) = M = N_2 \cap \mathrm{Hb}(P)$ where M belongs to $\mathrm{Sem}_{\mathcal{C}}(P)$. But then we obtain $N_1 = \mathrm{Ext}(M) = N_2$ by the third requirement on Tr, a contradiction. Thus Proj is injective as well.

In fact, the requirements on Tr imply that $\mathrm{Proj}(\mathrm{Ext}(M)) = M$ for any $M \in \mathrm{Sem}_{\mathcal{P}}(P)$. Thus Ext and Proj are inverses of each other as well as bijective. In addition, we have $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$ by the first requirement on Tr. Let M be any model from $\mathrm{Sem}_{\mathcal{C}}(P)$. Recall that $M = \mathrm{Ext}(M) \cap \mathrm{Hb}(P)$ holds by the second requirement on Tr. Since $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P)) \subseteq \mathrm{Hb}(P)$, we obtain $M \cap \mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Ext}(M) \cap \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P))$. Since the choice of M was arbitrary, we may conclude $P \equiv_{\mathrm{v}} \mathrm{Tr}(P)$. □

4 EXPRESSIVE POWER ANALYSIS

In this section, we compare the expressive powers of the classes of logic programs introduced in Section 2.3 using the classification method presented in Section 3.4. Due to the nature of the syntactic constraints imposed in Section 2.3, the key problem is to see whether there are ways to reduce the number of positive body literals in the bodies of rules. The results of this section will indicate that this is possible to some extent, but not in general, i.e. there is no faithful and modular way of removing *all* positive body literals from rules. As a preparation for the expressive power analysis, we distinguish certain properties program modules in Section 4.1. The actual analysis takes place in two phases. At first, we address the class of positive programs $\mathcal{P}^+$ and the respective subclasses in Section 4.2. After that the class normal programs $\mathcal{P}$ is subjected to similar analysis in Section 4.3. However, the extended syntax of normal logic programs and the stable model semantics in its full generality makes the analysis more intricate and involved. Finally, we take classical propositional logic into consideration in Section 4.4 and relate sets of clauses under classical models with the other classes.

4.1 Some Properties of Programs Modules

We prepare the forthcoming expressive power analysis by presenting two useful properties of program modules under the least/stable model semantics. The first property is related with a positive program $P \cup Q$ consisting of two subprograms P and Q so that the module conditions M1–M4 from Definition 3.15 are satisfied. Here the aim is to provide sufficient conditions for removing either one of the modules by evaluating its effect on the joint least model $\mathrm{LM}(P \cup Q)$ and by replacing it with a compensating atomic program. Formally, we propose a reduction that yields a set of atomic rules.

Definition 4.1 *Given a positive normal program $P \in \mathcal{P}^+$ and an interpretation I, the* ***visible net reduction*** *of P is $P_I^{\mathrm{v}} = \{\mathsf{a} \leftarrow \mid \mathsf{a} \in \mathrm{Hb}_{\mathrm{v}}(P) \cap I\}$ so that $\mathrm{Hb}_{\mathrm{v}}(P_I^{\mathrm{v}}) = \mathrm{Hb}(P_I^{\mathrm{v}}) = \mathrm{Hb}_{\mathrm{v}}(P)$ which makes all atoms of P_I^{v} visible.*

The reduced program P_I^{v} overestimates P in a sense, since $\mathsf{a} \leftarrow$ may be included in P_I^{v} even if there is no rule $r \in P$ such that $\mathrm{head}(r) = \mathsf{a}$. However, the reduct P_I^{v} can be formed externally without knowing exactly which rules constitute the program P being reduced. In addition, we assumed that the interpretation I in Definition 4.1 may contain atoms outside $\mathrm{Hb}(P)$. This setting is easily realized when P is placed as a program module in the context of another program Q. If I is a model for $P \cup Q$, then the set of atoms encoded as atomic rules in P_I^{v} can be understood as the maximum contribution of the rules of P for the atoms that true in I. In the sequel, we will apply the visible net reduction w.r.t. least models in the following way.

Lemma 4.2 *Let P and Q be two positive programs satisfying the module conditions M1–M4 and $M = \mathrm{LM}(P \cup Q) \subseteq \mathrm{Hb}(P) \cup \mathrm{Hb}(Q)$. Then $\mathrm{LM}(P_M^{\mathrm{v}} \cup Q) = M \cap (\mathrm{Hb}_{\mathrm{v}}(P) \cup \mathrm{Hb}(Q))$ holds for the visible net reduct P_M^{v}.*

PROOF. Let us define $N = M \cap (\mathrm{Hb_v}(P) \cup \mathrm{Hb}(Q))$ and assume that $N \not\models P^{\mathrm{v}}_M \cup Q$. This leads to two possibilities: $N \not\models P^{\mathrm{v}}_M$ or $N \not\models Q$.

(i) If $N \not\models P^{\mathrm{v}}_M$, then we know by Definition 4.1 that there is a rule $\mathsf{a} \leftarrow$ in P^{v}_M such that $\mathsf{a} \in \mathrm{Hb_v}(P)$, $\mathsf{a} \in M$, and $N \not\models \mathsf{a} \leftarrow$. The last implies $\mathsf{a} \notin N$ and $\mathsf{a} \notin M$ by the definition of N, a contradiction.

(ii) If $N \not\models Q$, there is a rule $r \in Q$ such that $N \not\models r$, i.e. $\mathrm{head}(r) \notin N$ and $\mathrm{body}^+(r) \subseteq N$. It follows by the definition of N that $\mathrm{head}(r) \notin M$ and $\mathrm{body}^+(r) \subseteq M$, as $\mathrm{head}(r) \in \mathrm{Hb}(Q)$ and $\mathrm{body}^+(r) \subseteq \mathrm{Hb}(Q)$. But then we have $M \not\models r$. A contradiction, as $M \models Q$.

Thus $N \models P^{\mathrm{v}}_M \cup Q$ holds and it follows that $\mathrm{LM}(P^{\mathrm{v}}_M \cup Q) \subseteq N$.

To establish the converse inclusion, we define $M_i = \mathrm{T}_{P \cup Q} \uparrow i$ and $N_i = M_i \cap (\mathrm{Hb_v}(P) \cup \mathrm{Hb}(Q))$ for all $i \geq 0$ and prove $N_i \subseteq \mathrm{LM}(P^{\mathrm{v}}_M \cup Q)$ by induction on i. The base case is trivial, as $N_i = M_i = \emptyset$ by definition. Let us then assume that $\mathsf{a} \in N_i$, i.e. $\mathsf{a} \in M_i$ and $\mathsf{a} \in \mathrm{Hb_v}(P) \cup \mathrm{Hb}(Q)$. Then there is a rule $r \in P \cup Q$ such that $\mathrm{head}(r) = \mathsf{a}$ and $\mathrm{body}^+(r) \subseteq M_{i-1}$. If $\mathsf{a} \in N_{i-1}$ holds, too, then $\mathsf{a} \in \mathrm{LM}(P^{\mathrm{v}}_M \cup Q)$ by the inductive hypothesis directly. Thus $\mathsf{a} \notin N_{i-1}$ is assumed in the sequel. Two cases have to be considered; recall that $P \cap Q = \emptyset$ by the module condition M1.

(a) If $r \in Q$, we know that $\mathrm{head}(r) \in \mathrm{Hb}(Q)$ and $\mathrm{body}^+(r) \subseteq \mathrm{Hb}(Q)$. Thus $\mathrm{body}^+(r) \subseteq N_{i-1}$ which is included in $\mathrm{LM}(P^{\mathrm{v}}_M \cup Q)$ by the inductive hypothesis. Consequently, we know that $\mathrm{head}(r) = \mathsf{a} \in \mathrm{LM}(P^{\mathrm{v}}_M \cup Q)$.

(b) If $r \in P$, it holds that $\mathrm{head}(r) \in \mathrm{Hb_v}(P)$ and $\mathrm{body}^+(r) \subseteq \mathrm{Hb}(P)$, as $\mathrm{head}(r) = \mathsf{a} \in \mathrm{Hb_v}(P)$. Thus the rule $\mathrm{head}(r) \leftarrow \in P^{\mathrm{v}}_M$. It follows that $\mathrm{head}(r) = \mathsf{a}$ is contained in $\mathrm{LM}(P^{\mathrm{v}}_M \cup Q)$.

Thus $\mathsf{a} \in \mathrm{LM}(P^{\mathrm{v}}_M \cup Q)$ holds unconditionally and we have established that $N_i \subseteq \mathrm{LM}(P^{\mathrm{v}}_M \cup Q)$ for all $i \geq 0$. It follows that $N \subseteq \mathrm{LM}(P^{\mathrm{v}}_M \cup Q)$. □

The second property lets us to combine stable models of program modules under certain circumstances to form a stable model for a larger program.

Lemma 4.3 *Let P and Q be two normal programs satisfying the module conditions. If $M \in \mathrm{SM}(P)$, $N \in \mathrm{SM}(Q)$, and $M \cap \mathrm{Hb_v}(P) \cap \mathrm{Hb_v}(Q) = N \cap \mathrm{Hb_v}(P) \cap \mathrm{Hb_v}(Q)$, then $M \cup N \in \mathrm{SM}(P \cup Q)$ and $(P \cup Q)^{N \cup M} = P^N \cup Q^M$.*

PROOF. Let $M \in \mathrm{SM}(P)$ and $N \in \mathrm{SM}(Q)$ such that $M \cap \mathrm{Hb_v}(P) \cap \mathrm{Hb_v}(Q) = N \cap \mathrm{Hb_v}(P) \cap \mathrm{Hb_v}(Q)$. Let us prove $(P \cup Q)^{M \cup N} = P^M \cup Q^N$.

Consider any rule $r \in P$ and any $\mathsf{a} \in \mathrm{body}^-(r)$. If $\mathsf{a} \in N \subseteq \mathrm{Hb}(Q)$, it follows that $\mathsf{a} \in \mathrm{Hb_v}(P) \cap \mathrm{Hb_v}(Q)$, since P and Q satisfy the module conditions. Thus also $\mathsf{a} \in M$ by our assumptions on N and M. Thus $\mathrm{body}^-(r) \cap N \neq \emptyset$ implies $\mathrm{body}^-(r) \cap M \neq \emptyset$. It follows that $r^+ \in P^{M \cup N} \iff \mathrm{body}^-(r) \cap (M \cup N) = \emptyset \iff \mathrm{body}^-(r) \cap M = \emptyset$ and $\mathrm{body}^-(r) \cap N = \emptyset \iff \mathrm{body}^-(r) \cap M = \emptyset \iff r^+ \in P^M$. Thus we have established that $P^{M \cup N} = P^M$ and $Q^{M \cup N} = Q^N$ is obtained by symmetry. All that remains to be noted is that $(P \cup Q)^{M \cup N} = P^{M \cup N} \cup Q^{M \cup N}$ holds directly by definition.

A direct consequence is that $M \cup N = \mathrm{LM}(P^M) \cup \mathrm{LM}(Q^N)$ is contained in $\mathrm{LM}(P^M \cup Q^N) = \mathrm{LM}((P \cup Q)^{M \cup N})$. To establish the inclusion in the other direction, let us assume that $M \cup N \not\models P^M \cup Q^N$. Two cases arise.

(i) If $M \cup N \not\models P^M$ holds, there is a rule $r \in P$ such that $\mathrm{body}^-(r) \cap M = \emptyset$, $\mathrm{body}^+(r) \subseteq M \cup N$, and $\mathrm{head}(r) \notin M \cup N$. Now any atom $\mathsf{a} \in \mathrm{body}^+(r) \subseteq \mathrm{Hb}(P)$ that is contained in $N \subseteq \mathrm{Hb}(Q)$ is also contained in $M \subseteq \mathrm{Hb}(P)$. This is because any such atom must be a member of $\mathrm{Hb}_{\mathrm{v}}(P) \cap \mathrm{Hb}_{\mathrm{v}}(Q)$ and the models M and N are assumed to coincide on these atoms. It follows that $\mathrm{body}^+(r) \subseteq M$ and $\mathrm{head}(r) \notin M$. But this means that $M \not\models P^M$ is the case, a contradiction.

(ii) If $M \cup N \not\models Q^N$ holds, we obtain a contradiction by symmetry.

Therefore $M \cup N \models P^M \cup Q^N$ is the case which implies $\mathrm{LM}(P^M \cup Q^N) \subseteq M \cup N$. Thus $M \cup N = \mathrm{LM}(P^M \cup Q^N) = \mathrm{LM}((P \cup Q)^{M \cup N})$. □

4.2 Positive Programs

In Section 2.3, we identified three subclasses of $\mathcal{P}^+$ which are obtained by restricting the syntax of the rules whereas the semantics of logic programs in these classes remains unchanged. Thus we obtain the relationships $\mathcal{A}^+ \leq_{\mathrm{PFM}} \mathcal{U}^+ \leq_{\mathrm{PFM}} \mathcal{B}^+ \leq_{\mathrm{PFM}} \mathcal{P}^+$ directly by Proposition 3.18, but it remains open whether these relationships are strict or not. We begin with a study of the relationship $\mathcal{B}^+ \leq_{\mathrm{PFM}} \mathcal{P}^+$. In fact, any *non-binary* rule $\mathsf{a} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_n$ where $n > 2$ can be rewritten to reduce the number of atoms that appear in the body of the rule. One technique is to introduce $n-1$ new atoms $\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}$, which remain local to this particular rule, and the following binary rules:

$$\begin{array}{ll} & \mathsf{a} \leftarrow \mathsf{b}_1, \mathsf{a}_1; \\ (4.1) & \mathsf{a}_1 \leftarrow \mathsf{b}_2, \mathsf{a}_2;\ \ldots;\ \mathsf{a}_{n-2} \leftarrow \mathsf{b}_{n-1}, \mathsf{a}_{n-1}; \\ & \mathsf{a}_{n-1} \leftarrow \mathsf{b}_n. \end{array}$$

It should be intuitively clear that the head a of the original rule can be inferred using these binary rules whenever the body atoms $\mathsf{b}_1, \ldots, \mathsf{b}_n$ are all inferable. As a result, any non-binary program P gets translated into a binary one $\mathrm{Tr}_{\mathrm{BIN}}(P)$ so that the corresponding translation function is PFM.

Theorem 4.4 $\mathcal{P}^+ \leq_{\mathrm{PFM}} \mathcal{B}^+$.

PROOF. A special case of the proof of Theorem 4.16. □

Corollary 4.5 $\mathcal{B}^+ =_{\mathrm{PFM}} \mathcal{P}^+$.

Let us then address the strictness of the relationship $\mathcal{U}^+ \leq_{\mathrm{PFM}} \mathcal{B}^+$. It turns out in the sequel that there is no PFM translation function from $\mathcal{B}^+$ to $\mathcal{U}^+$. To establish this, we need a subsidiary result on *unary programs* P: if an atom a is included in $\mathrm{LM}(P)$, then P contains at least one atomic rule which causes the atom a to be inferable by the strictly unary rules of P, i.e. to be included in $\mathrm{LM}(P)$. Note that $\mathrm{LM}(P) = \emptyset$ for any strictly unary program P.

Lemma 4.6 *Let $P = P_1 \cup P_0$ be a unary positive program where P_1 contains the strictly unary rules of P and P_0 contains the atomic rules of P. If $\mathsf{a} \in \mathrm{LM}(P)$, then P_0 contains an atomic rule $\mathsf{b} \leftarrow$ such that $\mathsf{a} \in \mathrm{LM}(P_1 \cup \{\mathsf{b} \leftarrow\})$.*

PROOF. We use complete induction on the level number $\mathrm{lev}(\mathsf{a}) > 0$ to prove the claim for any atom $\mathsf{a} \in \mathrm{LM}(P_1 \cup P_0) = \mathrm{LM}(P)$.

For the base case, assume that $\mathrm{lev}(\mathsf{a}) = 1$ which implies that $\mathsf{a} \in \mathrm{T}_{P_1 \cup P_0} \uparrow 1 = \mathrm{T}_{P_1 \cup P_0}(\emptyset)$. Thus $\mathsf{a} \leftarrow$ must appear in P_0. It is also clear that $\mathsf{a} \in \mathrm{T}_{P_1 \cup \{\mathsf{a} \leftarrow\}} \uparrow 1 = \mathrm{T}_{P_1 \cup \{\mathsf{a} \leftarrow\}}(\emptyset)$ so that $\mathsf{a} \in \mathrm{LM}(P_1 \cup \{\mathsf{a} \leftarrow\})$.

Then consider the case that $\mathrm{lev}(\mathsf{a}) = i > 1$. Then $\mathsf{a} \in \mathrm{T}_{P_1 \cup P_0} \uparrow i$ and there is a unary rule $\mathsf{a} \leftarrow \mathsf{b} \in P_1$ such that $\mathsf{b} \in \mathrm{T}_{P_1 \cup P_0} \uparrow i - 1$. Since $\mathsf{b} \in \mathrm{LM}(P_1 \cup P_0)$ and $0 < \mathrm{lev}(\mathsf{b}) < \mathrm{lev}(\mathsf{a}) = i$, it follows by the inductive hypothesis that there is an atomic rule $\mathsf{c} \leftarrow$ in P_0 such that $\mathsf{b} \in \mathrm{LM}(P_1 \cup \{\mathsf{c} \leftarrow\})$. This implies $\mathsf{a} \in \mathrm{LM}(P_1 \cup \{\mathsf{c} \leftarrow\})$, because the rule $\mathsf{a} \leftarrow \mathsf{b}$ belongs to P_1. □

We are now ready to establish that unary programs are strictly less expressive than binary ones. The proof below demonstrates how it is impossible to express the conjunctive condition (b and c) in the body of a rule $\mathsf{a} \leftarrow \mathsf{b}, \mathsf{c}$ using unary rules. In fact, if we attempt to capture this condition in terms of unary rules, the condition turns into a disjunctive one: already b or c alone is sufficient to infer a — assuming that a does not follow from our translation directly. It is also worth pointing out that our counter-example does not depend on the polynomiality requirement. Thus the intranslatability result embodied in Theorem 4.7 covers arbitrarily large translations!

Theorem 4.7 $\mathcal{B}^+ \not\leq_{\mathrm{FM}} \mathcal{U}^+$.

PROOF. Let us assume that there is a faithful and modular translation function $\mathrm{Tr}_{\mathrm{UN}}$ from $\mathcal{B}^+$ to $\mathcal{U}^+$. Then consider a strictly binary program $B = \{\mathsf{a} \leftarrow \mathsf{b}, \mathsf{c}\}$, atomic (as well as unary) programs $A_1 = \{\mathsf{b} \leftarrow\}$ and $A_2 = \{\mathsf{c} \leftarrow\}$, and the translation $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2)$. Note that B, A_1, and A_2 satisfy the module conditions as any combination, since $\mathrm{Hb}_{\mathrm{v}}(B) = \mathrm{Hb}(B) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $\mathrm{Hb}_{\mathrm{v}}(A_1) = \mathrm{Hb}(A_1) = \{\mathsf{b}\}$, and $\mathrm{Hb}_{\mathrm{v}}(A_2) = \mathrm{Hb}(A_2) = \{\mathsf{c}\}$. Consequently, the modularity of $\mathrm{Tr}_{\mathrm{UN}}$ implies that $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2) = \mathrm{Tr}_{\mathrm{UN}}(B) \cup \mathrm{Tr}_{\mathrm{UN}}(A_1) \cup \mathrm{Tr}_{\mathrm{UN}}(A_2)$ where the modules $\mathrm{Tr}_{\mathrm{UN}}(B)$, $\mathrm{Tr}_{\mathrm{UN}}(A_1)$, and $\mathrm{Tr}_{\mathrm{UN}}(A_2)$ are unary.

Since $\mathrm{Hb}_{\mathrm{v}}(B \cup A_1 \cup A_2) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and $M = \mathrm{LM}(B \cup A_1 \cup A_2) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, the faithfulness of $\mathrm{Tr}_{\mathrm{UN}}$ implies $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2)) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\} \subseteq N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2)) = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B) \cup \mathrm{Tr}_{\mathrm{UN}}(A_1) \cup \mathrm{Tr}_{\mathrm{UN}}(A_2))$. Similarly, we note that $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(B)) = \mathrm{Hb}_{\mathrm{v}}(B) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(A_1)) = \mathrm{Hb}_{\mathrm{v}}(A_1) = \{\mathsf{b}\}$, and $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(A_2)) = \mathrm{Hb}_{\mathrm{v}}(A_2) = \{\mathsf{c}\}$. Then we may apply Lemma 4.2 to obtain $(\mathrm{Tr}_{\mathrm{UN}}(A_1) \cup \mathrm{Tr}_{\mathrm{UN}}(A_2))^{\mathrm{v}}_N = \{\mathsf{b} \leftarrow;\ \mathsf{c} \leftarrow\}$ and

$$
\begin{aligned}
(4.2) \quad N' &= \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B) \cup \{\mathsf{b} \leftarrow;\ \mathsf{c} \leftarrow\}) \\
&= N \cap (\mathrm{Hb}(\mathrm{Tr}_{\mathrm{UN}}(B)) \cup \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(A_1)) \cup \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(A_2))) \\
&= N \cap \mathrm{Hb}(\mathrm{Tr}_{\mathrm{UN}}(B)).
\end{aligned}
$$

Let us then partition the translation $\mathrm{Tr}_{\mathrm{UN}}(B)$ into a set of atomic rules $\mathrm{Tr}_{\mathrm{UN}}(B)_0$ and a set of strictly unary rules $\mathrm{Tr}_{\mathrm{UN}}(B)_1$. Since $\mathsf{a} \in N'$, we

may use Lemma 4.6 to establish that $\mathrm{Tr}_{\mathrm{UN}}(B)_0 \cup \{\mathsf{b} \leftarrow;\ \mathsf{c} \leftarrow\}$ contains an atomic rule $\mathsf{d} \leftarrow$ such that $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)_1 \cup \{\mathsf{d} \leftarrow\})$. There are three possibilities: either $\mathsf{d} = \mathsf{b}$, $\mathsf{d} = \mathsf{b}$, or $\mathsf{d} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{UN}}(B)_0$. Each of these three cases implies $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B) \cup \{\mathsf{b} \leftarrow\})$ *or* $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B) \cup \{\mathsf{c} \leftarrow\})$.

On the other hand, the programs A_1 and A_2 give rise to $M_1 = \mathrm{LM}(A_1) = \{\mathsf{b}\}$ and $M_2 = \mathrm{LM}(A_2) = \{\mathsf{c}\}$. Then the corresponding models $N_1 = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(A_1))$ and $N_2 = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(A_2))$ satisfy $\mathsf{b} \in N_1$ and $\mathsf{c} \in N_2$, respectively, as $\mathrm{Tr}_{\mathrm{UN}}$ is faithful. It follows by the monotonicity of the operator $\mathrm{LM}(\cdot)$ that $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B) \cup \mathrm{Tr}_{\mathrm{UN}}(A_1))$ or $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B) \cup \mathrm{Tr}_{\mathrm{UN}}(A_2))$. Thus $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1))$ holds or $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B \cup A_2))$ holds by the modularity of $\mathrm{Tr}_{\mathrm{UN}}$. But then $\mathsf{a} \in \mathrm{LM}(B \cup A_1)$ or $\mathsf{a} \in \mathrm{LM}(B \cup A_2)$, because $\mathrm{Tr}_{\mathrm{UN}}$ is faithful. A contradiction, since $\mathrm{LM}(B \cup A_1) = \mathrm{LM}(B \cup A_2) = \emptyset$. □

Corollary 4.8 *$\mathcal{B}^+ \not\leq_{\mathrm{PFM}} \mathcal{U}^+$, $\mathcal{U}^+ <_{\mathrm{PFM}} \mathcal{B}^+$, and $\mathcal{U}^+ <_{\mathrm{PFM}} \mathcal{P}^+$.*

An intranslatability result is obtained in the last case as well.

Lemma 4.9 *For positive atomic programs P, $\mathrm{LM}(P) = \{\mathrm{head}(r) \mid r \in P\}$.*

Theorem 4.10 *$\mathcal{U}^+ \not\leq_{\mathrm{FM}} \mathcal{A}^+$.*

PROOF. Let us suppose that there is a faithful and modular translation function $\mathrm{Tr}_{\mathrm{AT}}$ from $\mathcal{U}^+$ to $\mathcal{A}^+$. Let us then study a strictly unary program $U = \{\mathsf{a} \leftarrow \mathsf{b}\}$, an atomic program $A = \{\mathsf{b} \leftarrow\}$, and the translation $\mathrm{Tr}_{\mathrm{AT}}(U \cup A)$. Since $\mathrm{Hb}_{\mathrm{v}}(U) = \mathrm{Hb}(U) = \{\mathsf{a}, \mathsf{b}\}$ and $\mathrm{Hb}_{\mathrm{v}}(A) = \mathrm{Hb}(A) = \{\mathsf{b}\}$, the module conditions are trivially satisfied by U and A. Then the modularity of $\mathrm{Tr}_{\mathrm{AT}}$ implies $\mathrm{Tr}_{\mathrm{AT}}(U \cup A) = \mathrm{Tr}_{\mathrm{AT}}(U) \cup \mathrm{Tr}_{\mathrm{AT}}(A)$ where both $\mathrm{Tr}_{\mathrm{AT}}(U)$ and $\mathrm{Tr}_{\mathrm{AT}}(A)$ are atomic programs. Let us then appeal to the faithfulness of $\mathrm{Tr}_{\mathrm{AT}}$: $\mathsf{a} \in \mathrm{LM}(U \cup A)$ and $\mathsf{a} \notin \mathrm{LM}(A)$ imply $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U) \cup \mathrm{Tr}_{\mathrm{AT}}(A))$ and $\mathsf{a} \notin \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(A))$, respectively. It follows by Lemma 4.9 that the rule $\mathsf{a} \leftarrow$ must appear in $\mathrm{Tr}_{\mathrm{AT}}(U)$, since both $\mathrm{Tr}_{\mathrm{AT}}(U)$ and $\mathrm{Tr}_{\mathrm{AT}}(A)$ are atomic.

On the other hand, we have $\mathrm{LM}(U) = \emptyset$ which implies $\mathsf{a} \notin \mathrm{LM}(U)$. It follows by the faithfulness of $\mathrm{Tr}_{\mathrm{AT}}$ that $\mathsf{a} \notin \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U))$, a contradiction. □

Corollary 4.11 *The class $\mathcal{A}^+$ relates to other classes as follows:*

$$\mathcal{U}^+ \not\leq_{\mathrm{PFM}} \mathcal{A}^+,\ \mathcal{A}^+ <_{\mathrm{PFM}} \mathcal{U}^+,\ \mathcal{A}^+ <_{\mathrm{PFM}} \mathcal{B}^+,\ \text{and}\ \mathcal{A}^+ <_{\mathrm{PFM}} \mathcal{P}^+.$$

Theorems 4.4, 4.7 and 4.10 establish a strict ordering among the classes of logic programs $\mathcal{A}^+, \mathcal{U}^+, \mathcal{B}^+$ and $\mathcal{P}^+$ that can be summarized by a hierarchy:

$$\mathcal{A}^+ <_{\mathrm{PFM}} \mathcal{U}^+ <_{\mathrm{PFM}} \mathcal{B}^+ =_{\mathrm{PFM}} \mathcal{P}^+. \tag{4.3}$$

The relationships in this hierarchy indicate that the number of positive body literals can be limited to two without an effective loss of expressive power.

4.3 Normal Programs

The analysis of normal programs resembles that of positive programs. By the mutual inclusions of the classes and Proposition 3.18, we obtain analogous relationships $\mathcal{A} \leq_{\mathrm{PFM}} \mathcal{U} \leq_{\mathrm{PFM}} \mathcal{B} \leq_{\mathrm{PFM}} \mathcal{P}$. From now on, our plan is to generalize Theorems 4.4, 4.7 and 4.10 to the case of normal logic programs. Realizing such a plan boils down to establishing that negative body literals do not compensate positive body literals in a way that would make a difference with respect to the case of positive programs. As the first step of the plan, we generalize the translation function $\mathrm{Tr}_{\mathrm{BIN}}$ for normal programs. A single rule r given in (2.1) is translated into the following set of rules $\mathrm{Tr}_{\mathrm{BIN}}(r)$:

$$\begin{array}{ll} & \mathsf{a} \leftarrow \mathsf{b}_1, \mathsf{a}_1, {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m; \\ (4.4) & \mathsf{a}_1 \leftarrow \mathsf{b}_2, \mathsf{a}_2;\ \ldots;\ \mathsf{a}_{n-2} \leftarrow \mathsf{b}_{n-1}, \mathsf{a}_{n-1}; \\ & \mathsf{a}_{n-1} \leftarrow \mathsf{b}_n \end{array}$$

where $\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}$ are new atoms that are supposed to remain local to r. The reader might find it tempting to copy negative body literals ${\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m$ to every rule in (4.4), but a quadratic translation would result.

Definition 4.12 *For every $P \in \mathcal{P}$, define* $\mathrm{Tr}_{\mathrm{BIN}}(P) =$

$$\{r \mid r \in P \text{ and } |\mathrm{body}^+(r)| \leq 2\} \cup \bigcup \{\mathrm{Tr}_{\mathrm{BIN}}(r) \mid r \in P \text{ and } |\mathrm{body}^+(r)| > 2\}. \tag{4.5}$$

Moreover, let $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{BIN}}(P)) = \mathrm{Hb}_{\mathrm{a}}(P)$ *and* $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{BIN}}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$.

To ease correctness considerations, we define for each non-binary rule $r \in P$, the translation $\mathrm{Tr}_{\mathrm{BIN}}(r)$ in (4.4), and an interpretation $I \subseteq \mathrm{Hb}(P)$, the set of *implied body atoms* $\mathrm{IBA}(r, I)$ which contains a_i from (4.4) whenever $0 < i < n$ and $\mathsf{b}_{i+1} \in I, \ldots, \mathsf{b}_n \in I$. For binary rules $r \in P$, $\mathrm{IBA}(r, I) = \emptyset$. Then we may define $\mathrm{IBA}(P, I) = \bigcup\{\mathrm{IBA}(r, I) \mid r \in P\}$ for a normal program P and an interpretation $I \subseteq \mathrm{Hb}(P)$. Note that the Herbrand base $\mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(P))$ is obtained as $\mathrm{Hb}(P) \cup \mathrm{IBA}(P, \mathrm{Hb}(P))$.

Lemma 4.13 *Let P be a normal program and Q the translation* $\mathrm{Tr}_{\mathrm{BIN}}(P)$.

1. *If $M_1 \subseteq M_2 \subseteq \mathrm{Hb}(P)$ and $M_1 \models P^{M_2}$, then $N_1 \models \mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$ where $N_1 = M_1 \cup \mathrm{IBA}(P, M_1)$ and $N_2 = M_2 \cup \mathrm{IBA}(P, M_2)$.*

2. *If $N_1 \subseteq N_2 \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(P))$ and $N_1 \models \mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$, then $M_1 \subseteq M_2$ and $M_1 \models P^{M_2}$ hold for $M_1 = N_1 \cap \mathrm{Hb}(P)$ and $M_2 = N_2 \cap \mathrm{Hb}(P)$.*

PROOF. (1) Suppose that $M_1 \subseteq M_2 \subseteq \mathrm{Hb}(P)$ and $M_1 \models P^{M_2}$. Define N_1 and N_2 as above. Then assume that $N_1 \not\models \mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$. Two cases arise.

(a) There is a rule $r \in P$ such that $|\mathrm{body}^+(r)| \leq 2$, r is included in $\mathrm{Tr}_{\mathrm{BIN}}(P)$, $N_2 \cap \mathrm{body}^-(r) = \emptyset$, and $N_1 \not\models r^+$. Since r contains only atoms from $\mathrm{Hb}(P)$, it follows that $M_2 \cap \mathrm{body}^-(r) = \emptyset$ and $M_1 \not\models r^+$. Thus $M_1 \not\models P^{M_2}$, contradicting our assumptions.

(b) There is a rule $r \in P$ such that $|\mathrm{body}^+(r)| > 2$, $\mathrm{Tr}_{\mathrm{BIN}}(r)$ from (4.4) is included in $\mathrm{Tr}_{\mathrm{BIN}}(P)$, and some rule of $\mathrm{Tr}_{\mathrm{BIN}}(r)^{N_2}$ is not satisfied in N_1. Basically, there are three different kind of rules in $\mathrm{Tr}_{\mathrm{BIN}}(r)$ giving rise to the respective cases. (i) Suppose that the rule involved is $\mathsf{a} \leftarrow \mathsf{b}_1, \mathsf{a}_1, {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m$ from $\mathrm{Tr}_{\mathrm{BIN}}(r)$. It follows that $N_2 \cap \{\mathsf{c}_1, \ldots, \mathsf{c}_m\} = \emptyset$, $\mathsf{b}_1 \in N_1$, $\mathsf{a}_1 \in N_1$, and $\mathsf{a} \notin N_1$. All atoms of these except a_1 is included in $\mathrm{Hb}(P)$. Consequently, we have $M_2 \cap \{\mathsf{c}_1, \ldots, \mathsf{c}_m\} = \emptyset$, $\mathsf{b}_1 \in M_1$, and $\mathsf{a} \notin M_1$. On the other hand, the definition of N_1 and the fact that $\mathsf{a}_1 \in N_1$ imply that $\mathsf{b}_2 \in M_1$, …, $\mathsf{b}_n \in M_1$. Thus $\{\mathsf{b}_1, \ldots, \mathsf{b}_n\} \subseteq M_1$ which implies that $M_1 \not\models r^+$. Since $M_2 \cap \{\mathsf{c}_1, \ldots, \mathsf{c}_m\} = \emptyset$, r^+ belongs to P^{M_2}, a contradiction with $M_1 \models P^{M_2}$. (ii) The unsatisfied rule is $\mathsf{a}_{i-1} \leftarrow \mathsf{b}_i, \mathsf{a}_i$ where $1 < i < n$. Note that this rule is included unconditionally in $\mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$. It follows that $\mathsf{b}_i \in N_1$, $\mathsf{a}_i \in N_1$, but $\mathsf{a}_{i-1} \notin N_1$. Using the definition of N_1 again, $\mathsf{a}_i \in N_1$ implies $\{\mathsf{b}_{i+1}, \ldots, \mathsf{b}_n\} \subseteq M_1$, and $\mathsf{a}_{i-1} \notin N_1$ implies $\{\mathsf{b}_i, \ldots, \mathsf{b}_n\} \not\subseteq M_1$. Thus $\mathsf{b}_i \notin M_1$ must be the case. It follows by the definition of N_1 that $\mathsf{b}_i \notin N_1$, a contradiction. (iii) As a special case, the rule in question is $\mathsf{a}_{n-1} \leftarrow \mathsf{b}_n$. It follows that $\mathsf{b}_n \in N_1$ but $\mathsf{a}_{n-1} \notin N_1$, a contradiction with the definition of N_1.

To conclude, $N_1 \not\models \mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$ implies a contradiction. Hence the claim.

(2) Let $N_1 \subseteq N_2 \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(P))$ and $N_1 \models \mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$. Define M_1 and M_2 as above so that $M_1 \subseteq M_2$. Let us assume that $M_1 \not\models P^{M_2}$. Then there is a rule $r \in P$ such that $M_2 \cap \mathrm{body}^-(r) = \emptyset$ and $M_1 \not\models r^+$. Two cases arise for our further consideration.

(a) If $|\mathrm{body}^+(r)| \leq 2$, then r is included in $\mathrm{Tr}_{\mathrm{BIN}}(P)$ as such and the definitions of M_1 and M_2 as the respective projections of N_1 and N_2 w.r.t. $\mathrm{Hb}(P)$ imply $N_2 \cap \mathrm{body}^-(r) = \emptyset$ and $N_1 \not\models r^+$. Thus $N_1 \not\models \mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$, a contradiction with our assumptions.

(b) If $|\mathrm{body}^+(r)| > 2$, then $\mathrm{Tr}_{\mathrm{BIN}}(r)$ from (4.4) is included in $\mathrm{Tr}_{\mathrm{BIN}}(P)$ instead of r. Since $M_1 \not\models r^+$, we know that $\{\mathsf{b}_1, \ldots, \mathsf{b}_n\} \subseteq M_1$ and $\mathsf{a} \notin M_1$. We prove by decreasing induction on $i \leq n-1$ that a_i is included in N_1. In the base case $i = n-1$, we obtain $\mathsf{a}_{n-1} \in N_1$, since $\mathsf{a}_{n-1} \leftarrow \mathsf{b}_n \in \mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$ is satisfied in N_1 and $\mathsf{b}_n \in N_1$ holds by the definition of M_1 as $\mathsf{b}_n \in M$. Induction step follows, i.e. let $i < n-1$. Note that the rule $\mathsf{a}_i \leftarrow \mathsf{b}_{i+1}, \mathsf{a}_{i+1}$ is unconditionally included in $\mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$. Since $\mathsf{b}_{i+1} \in M_1$ we obtain $\mathsf{b}_{i+1} \in N_1$ by the definition of M_1. Moreover, $\mathsf{a}_{i+1} \in N_1$ follows by the inductive hypothesis. Then we obtain $\mathsf{a}_i \in N_1$, as $\mathsf{a}_i \leftarrow \mathsf{b}_{i+1}, \mathsf{a}_{i+1}$ is satisfied by N_1. To conclude, we have established that $\mathrm{IBA}(r, M_1) \subseteq N_1$ and $\mathsf{a}_1 \in N_1$ in particular.

Then recall that $M_2 \cap \mathrm{body}^-(r) = \emptyset$, i.e. $M_2 \cap \{\mathsf{c}_1, \ldots, \mathsf{c}_m\} = \emptyset$. It follows by the definition of M_2 that $\mathsf{a} \leftarrow \mathsf{b}_1, \mathsf{a}_1$ is a member of $\mathrm{Tr}_{\mathrm{BIN}}(P)^{N_2}$. Since $\mathsf{b}_1 \in N_1$, $\mathsf{a}_1 \in N_1$, and $\mathsf{a} \leftarrow \mathsf{b}_1, \mathsf{a}_1$ is necessarily satisfied by N_1, we obtain $\mathsf{a} \in N_1$. Thus $\mathsf{a} \in M_1$ by the definition of M_1, a contradiction.

Thus $M_1 \not\models P^{M_2}$ leads to a contradiction. Hence $M_1 \models P^{M_2}$ is the case. □

Proposition 4.14 *Let P be a normal logic program. If M is a stable model of P, then $N = M \cup \mathrm{IBA}(P, M)$ is a stable model of $\mathrm{Tr}_{\mathrm{BIN}}(P)$ such that $M = N \cap \mathrm{Hb}(P)$.*

PROOF. Let M be a stable model of P, i.e. $M = \mathrm{LM}(P^M)$. Since $M \models P^M$, we obtain by the first item of Lemma 4.13 that $N \models \mathrm{Tr}_{\mathrm{BIN}}(P)^N$ holds for $N = M \cup \mathrm{IBA}(P, M)$. Since $\mathrm{IBA}(P, M) \subseteq \mathrm{IBA}(P, \mathrm{Hb}(P))$ consists of new atoms not appearing in $\mathrm{Hb}(P)$, it is clear that $M = N \cap \mathrm{Hb}(P)$. Let us then suppose that N is not the least model of $\mathrm{Tr}_{\mathrm{BIN}}(P)^N$, i.e. there is $N' \subset N$ such that $N' \models \mathrm{Tr}_{\mathrm{BIN}}(P)^N$. Consider any $\mathsf{a} \in N - N'$. Two cases arise depending on the membership of a in $\mathrm{Hb}(P)$.

- If $\mathsf{a} \in \mathrm{Hb}(P)$, then we obtain by the second item in Lemma 4.13 that $M' = N' \cap \mathrm{Hb}(P) \subset M$ is a model of P^M. But this contradicts the stability of M! Thus $N' \cap \mathrm{Hb}(P) = N \cap \mathrm{Hb}(P)$ is necessarily the case.

- If $\mathsf{a} \in \mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(P)) - \mathrm{Hb}(P)$, then a is one of the new atoms a_i involved in (4.4) and $\mathsf{a}_i \in \mathrm{IBA}(P, M)$, as $\mathsf{a}_i \in N$. It follows by the definition of $\mathrm{IBA}(P, M)$ that $\mathsf{b}_{i+1} \in M, \ldots, \mathsf{b}_n \in M$. Since $N' \cap \mathrm{Hb}(P) = N \cap \mathrm{Hb}(P)$, as established above, we obtain $\mathsf{b}_{i+1} \in N', \ldots, \mathsf{b}_n \in N'$ by the definition of N. Since $N' \models \mathrm{Tr}_{\mathrm{BIN}}(P)^N$ and the positive rules of (4.4) are unconditionally included in $\mathrm{Tr}_{\mathrm{BIN}}(P)^N$, it follows inductively that $\mathsf{a}_{n-1} \in N'$, $\mathsf{a}_{n-2} \in N'$, ..., $\mathsf{a}_i \in N'$. Thus $\mathsf{a} \in N'$, a contradiction.

Hence there is no a in the difference $N - N'$. Then $N' = N$ must hold — contradicting our previous assumption. This implies $N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{BIN}}(P)^N)$, i.e. N is a stable model of $\mathrm{Tr}_{\mathrm{BIN}}(P)$. □

Proposition 4.15 *Let P be a normal logic program. If N is a stable model of $\mathrm{Tr}_{\mathrm{BIN}}(P)$, then $M = N \cap \mathrm{Hb}(P)$ is a stable model of P such that $N = M \cup \mathrm{IBA}(P, M)$.*

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{BIN}}(P)$, i.e. $N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{BIN}}(P)^N)$. Since $N \models \mathrm{Tr}_{\mathrm{BIN}}(P)^N$, we obtain by the second item in Lemma 4.13 that $M \models P^M$ holds for $M = N \cap \mathrm{Hb}(P)$. Let A be the set of atoms $N \cap (\mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(P)) - \mathrm{Hb}(P))$ so that $N = M \cup A$. It is proved next that $A = \mathrm{IBA}(P, M)$. We do this by by showing for each non-binary rule $r \in P$ and the corresponding translation $\mathrm{Tr}_{\mathrm{BIN}}(r)$ in (4.4) that $\mathsf{a}_i \in A$ if and only if $\mathsf{a}_i \in \mathrm{IBA}(P, M)$. We use decreasing induction on $i \leq n - 1$ as follows.

In the base case $i = n - 1$. Now $\mathsf{a}_{n-1} \in A \iff \mathsf{a}_{n-1} \in N$ by the definition of $A \iff \mathsf{a}_{n-1} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{BIN}}(P)^N)$, as N is stable, $\iff \mathsf{b}_n \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{BIN}}(P)^N)$, as the rule $\mathsf{a}_{n-1} \leftarrow \mathsf{b}_n$ is included in $\mathrm{Tr}_{\mathrm{BIN}}(P)^N$, $\iff \mathsf{b}_n \in M$ by the definition of $M \iff \mathsf{a}_{n-1} \in \mathrm{IBA}(r, M)$. Induction step follows: $0 < i < n - 1$. It follows that $\mathsf{a}_i \in A \iff \mathsf{a}_i \in N$ by the definition of $A \iff \mathsf{a}_i \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{BIN}}(P)^N)$, as N is stable, $\iff \mathsf{b}_{i+1}$ and a_{i+1} belong to $N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{BIN}}(P)^N)$, as $\mathrm{Tr}_{\mathrm{BIN}}(P)^N$ contains the rule $\mathsf{a}_i \leftarrow \mathsf{b}_{i+1}, \mathsf{a}_{i+1}$, $\iff \mathsf{b}_{i+1} \in M$ and $\mathsf{a}_{i+1} \in A$ by the definitions of M and $A \iff \mathsf{b}_{i+1} \in M$ and $\mathsf{a}_{i+1} \in \mathrm{IBA}(r, M)$ by the inductive hypothesis $\iff \mathsf{a}_i \in \mathrm{IBA}(r, M)$.

Thus we have established that $N = M \cup \mathrm{IBA}(P, M)$. Then suppose that M is not the least model of P^M, i.e. there is $M' \subset M$ such that $M' \models P^M$. It follows by the first item in Lemma 4.13 that $N' \models \mathrm{Tr}_{\mathrm{BIN}}(P)^N$ holds for $N' =$

$M' \cup \mathrm{IBA}(P, M')$. Since $M' \subset M$, we obtain $\mathrm{IBA}(P, M') \subseteq \mathrm{IBA}(P, M)$ and $N' \subset N$, which contradicts the stability of N. Thus $M = \mathrm{LM}(P^M)$ is necessarily the case and M is a stable model of P. □

Theorem 4.16 $\mathcal{P} \leq_{\mathrm{PFM}} \mathcal{B}$.

PROOF. Let us begin with the faithfulness of $\mathrm{Tr}_{\mathrm{BIN}}$. It is clear by Definition 4.12 that $\mathrm{Hb}(P) \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(P))$ and $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{BIN}}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$. By Proposition 4.14 there is an extension function $\mathrm{Ext}_{\mathrm{BIN}} : \mathrm{SM}(P) \to \mathrm{SM}(\mathrm{Tr}_{\mathrm{BIN}}(P))$ that maps $M \in \mathrm{SM}(P)$ into $N = \mathrm{Ext}_{\mathrm{BIN}}(M) = M \cup \mathrm{IBA}(P, M)$ included in $\mathrm{SM}(\mathrm{Tr}_{\mathrm{BIN}}(P))$ such that $M = N \cap \mathrm{Hb}(P)$. In addition to this, we know by Proposition 4.15 that if $N \in \mathrm{SM}(\mathrm{Tr}_{\mathrm{BIN}}(P))$, then $M = N \cap \mathrm{Hb}(P) \in \mathrm{SM}(P)$ and $N = \mathrm{Ext}_{\mathrm{BIN}}(M)$. Thus $\mathrm{Tr}_{\mathrm{BIN}}$ is faithful by Theorem 3.19.

To establish modularity of $\mathrm{Tr}_{\mathrm{BIN}}$, let P and Q be two normal programs such that the module conditions M1–M4 from Definition 3.15 are satisfied. It is obvious by Definition 4.12 that $\mathrm{Tr}_{\mathrm{BIN}}(P \cup Q) = \mathrm{Tr}_{\mathrm{BIN}}(P) \cup \mathrm{Tr}_{\mathrm{BIN}}(Q)$. Let us then address the module conditions. **(M1)** Suppose that $\mathrm{Tr}_{\mathrm{BIN}}(P)$ and $\mathrm{Tr}_{\mathrm{BIN}}(Q)$ share a rule r. Two cases arise.

1. Suppose that $r \in P$. Then $|\mathrm{body}^+(r)| \leq 2$ holds by the definition of $\mathrm{Tr}_{\mathrm{BIN}}(P)$. Moreover, it follows that $r \notin Q$, as $P \cap Q = \emptyset$ by the module conditions. It follows that $r \notin \mathrm{Tr}_{\mathrm{BIN}}(Q)$, a contradiction.

2. Suppose that $r \in \mathrm{Tr}_{\mathrm{BIN}}(r')$ for some non-binary rule $r' \in P$. It follows by the module conditions that $r' \notin Q$. This means that no rule from $\mathrm{Tr}_{\mathrm{BIN}}(r')$ is included in $\mathrm{Tr}_{\mathrm{BIN}}(Q)$, since these rules are uniquely determined by new atoms $\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}$ which depend on r', a contradiction.

It follows that $\mathrm{Tr}_{\mathrm{BIN}}(P) \cap \mathrm{Tr}_{\mathrm{BIN}}(Q) = \emptyset$. **(M2)** Because P and Q satisfy the second module condition, we know that $\mathrm{Hb}_{\mathrm{a}}(P) \cap \mathrm{Hb}_{\mathrm{a}}(Q) = \emptyset$. By Definition 4.12, these sets are preserved by $\mathrm{Tr}_{\mathrm{BIN}}$, i.e. $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{BIN}}(P)) = \mathrm{Hb}_{\mathrm{a}}(P)$ and $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{BIN}}(Q)) = \mathrm{Hb}_{\mathrm{a}}(Q)$. Thus $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{BIN}}(P)) \cap \mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{BIN}}(Q)) = \emptyset$. **(M3)** Let us then assume that $\mathrm{Hb}_{\mathrm{h}}(\mathrm{Tr}_{\mathrm{BIN}}(P))$ and $\mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(Q))$ share some atom a. Again, two cases arise.

1. Assume that $\mathsf{a} \in \mathrm{Hb}_{\mathrm{h}}(P)$. Since P and Q satisfy module conditions, we know that $\mathsf{a} \notin \mathrm{Hb}(Q)$. Since $\mathsf{a} \in \mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(Q))$, the atom a must be one of the new atoms $\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}$ associated with a non-binary rule $r \in Q$ with $|\mathrm{body}^+(r)| = n$. A contradiction by Definition 4.12, as $\mathsf{a} \in \mathrm{Hb}_{\mathrm{h}}(P)$ and P is also subject to translation as a module of $P \cup Q$.

2. Suppose that $\mathsf{a} \in \mathrm{Hb}_{\mathrm{h}}(\mathrm{Tr}_{\mathrm{BIN}}(P)) - \mathrm{Hb}_{\mathrm{h}}(P)$. Then a must be one of the new atoms $\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}$ associated with a non-binary rule $r \in P$ with $|\mathrm{body}^+(r)| = n$. If $\mathsf{a} \in \mathrm{Hb}(Q)$, then a is not new, a contradiction. If $\mathsf{a} \in \mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(Q)) - \mathrm{Hb}(Q)$, then a must be one of the new atoms associated with a non-binary rule $r' \in Q$ with $|\mathrm{body}^+(r')| > 2$. Such atoms are different by Definition 4.12, a contradiction.

Thus $\mathrm{Hb}_{\mathrm{h}}(\mathrm{Tr}_{\mathrm{BIN}}(P)) \cap \mathrm{Hb}(\mathrm{Tr}_{\mathrm{BIN}}(Q)) = \emptyset$. **(M4)** The last module condition follows by symmetry with respect to the preceding one.

By definition 4.12 and the modularity of $\mathrm{Tr}_{\mathrm{BIN}}$, the translation $\mathrm{Tr}_{\mathrm{BIN}}(P)$ of a normal program $P \in \mathcal{P}$ can be computed on a rule-by-rule basis. Moreover, the translation can be done in time linear to $||P||$, because (i) binary rules can be passed on unmodified and (ii) any non-binary rule (2.1) consisting of $2n + 3m + 2$ symbols is replaced by n rules (4.4) consisting of $(6+3m)+(n-2)\times 6+4 = 6n+3m-2$ symbols, (iii) the atoms in $\mathrm{Hb}_{\mathrm{a}}(P)$ remain intact. □

Corollary 4.17 $\mathcal{B} =_{\mathrm{PFM}} \mathcal{P}$.

The main intranslatability result of this work follows: it is established that binary rules are not expressible in terms of unary rules even if we allow arbitrary number of negative literals in the bodies of rules.

Theorem 4.18 $\mathcal{B} \not\leq_{\mathrm{FM}} \mathcal{U}$.

PROOF. Let us assume that there is a faithful and modular translation function $\mathrm{Tr}_{\mathrm{UN}}$ from binary normal logic programs to unary ones. This translation function is to be applied to a strictly binary normal logic program

$$B = \{\mathsf{a} \leftarrow \mathsf{b}, \mathsf{c};\ \ \mathsf{b} \leftarrow \mathsf{c}, \mathsf{a};\ \ \mathsf{c} \leftarrow \mathsf{a}, \mathsf{b}\}$$

in conjunction with atomic programs $A_1 = \{\mathsf{a} \leftarrow\}$, $A_2 = \{\mathsf{b} \leftarrow\}$, and $A_3 = \{\mathsf{c} \leftarrow\}$. For these programs, $\mathrm{Hb}_{\mathrm{v}}(B) = \mathrm{Hb}(B) = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $\mathrm{Hb}_{\mathrm{v}}(A_1) = \mathrm{Hb}(A_1) = \{\mathsf{a}\}$, $\mathrm{Hb}_{\mathrm{v}}(A_2) = \mathrm{Hb}(A_2) = \{\mathsf{b}\}$, and $\mathrm{Hb}_{\mathrm{v}}(A_3) = \mathrm{Hb}(A_3) = \{\mathsf{c}\}$. As there are no invisible atoms and the rules of the four programs are all distinct, the module conditions from Definition 3.15 are trivially satisfied.

Note that the rules of B essentially express that if any two of the atoms a, b, and c are inferable, the third one should be, too. Thus each of the programs $B \cup A_1 \cup A_2$, $B \cup A_2 \cup A_3$, and $B \cup A_3 \cup A_1$ has a unique stable model $M = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$. Since $\mathrm{Tr}_{\mathrm{UN}}$ is faithful and modular, there are respective unique stable models

$$\left\{\begin{array}{l} N_1 = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^{N_1} \cup \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_1} \cup \mathrm{Tr}_{\mathrm{UN}}(A_2)^{N_1}) \\ N_2 = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^{N_2} \cup \mathrm{Tr}_{\mathrm{UN}}(A_2)^{N_2} \cup \mathrm{Tr}_{\mathrm{UN}}(A_3)^{N_2}) \\ N_3 = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^{N_3} \cup \mathrm{Tr}_{\mathrm{UN}}(A_3)^{N_3} \cup \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_3}) \end{array}\right. \tag{4.6}$$

of the translations $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2)$, $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_2 \cup A_3)$, and $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_3 \cup A_1)$. These three stable models have to be assumed different, as the modules constituting the respective translations may involve invisible atoms and each of them is based on a different combination of modules.

Let us then turn our attention to the first equation in (4.6) and the "missing module" $\mathrm{Tr}_{\mathrm{UN}}(A_3)$. Note that A_3 has a unique stable model $M_3 = \{\mathsf{c}\}$. Let $N_3' = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(A_3)^{N_3'})$ be the corresponding unique stable model of $\mathrm{Tr}_{\mathrm{UN}}(A_3)$, as implied by the faithfulness of $\mathrm{Tr}_{\mathrm{UN}}$. Note that $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2)$ and $\mathrm{Tr}_{\mathrm{UN}}(A_3)$ satisfy the module conditions, as $B \cup A_1 \cup A_2$ and A_3 do and $\mathrm{Tr}_{\mathrm{UN}}$ is modular. In addition, $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2)) \cap \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(A_3)) = \{\mathsf{c}\}$ and both N_1 and N_3' contain c so that we may apply Lemma 4.3 to conclude that $N_1 \cup N_3'$ is a stable model of $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2) \cup \mathrm{Tr}_{\mathrm{UN}}(A_3)$ which equals to $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2 \cup A_3)$ by the modularity of $\mathrm{Tr}_{\mathrm{UN}}$. Moreover, the reduct of the translation w.r.t. $N_1 \cup N_3'$ is $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2)^{N_1} \cup \mathrm{Tr}_{\mathrm{UN}}(A_3)^{N_3'}$.

We let N_1' stand for the unique stable model of $\mathrm{Tr}_{\mathrm{UN}}(A_1)$ which is guaranteed to exist by symmetry and which corresponds to the unique stable model $M_1 = \{\mathsf{a}\}$ of A_1. Similarly, let N_2' be the unique stable model corresponding to $M_2 = \{\mathsf{b}\}$. Then we may conclude that also $N_2 \cup N_1'$ and $N_3 \cup N_2'$ are stable models of $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2 \cup A_3)$ by using the last two equations of 4.6 concerning N_2 and N_3 in a symmetric fashion. On the other hand, M is the unique stable model of $B \cup A_1 \cup A_2 \cup A_3$, too. But then $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2 \cup A_3)$ must have a unique stable model — implying that $N_1 \cup N_3' = N_2 \cup N_1' = N_3 \cup N_2'$. Thus we may distinguish $N = N_1 \cap \mathrm{Hb}(\mathrm{Tr}_{\mathrm{UN}}(B)) = N_2 \cap \mathrm{Hb}(\mathrm{Tr}_{\mathrm{UN}}(B)) = N_3 \cap \mathrm{Hb}(\mathrm{Tr}_{\mathrm{UN}}(B))$, and rewrite the preceding set of equalities as

$$(4.7) \quad \begin{cases} N \cup N_1' \cup N_2' = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_1'} \cup \mathrm{Tr}_{\mathrm{UN}}(A_2)^{N_2'}) \\ N \cup N_2' \cup N_3' = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \mathrm{Tr}_{\mathrm{UN}}(A_2)^{N_2'} \cup \mathrm{Tr}_{\mathrm{UN}}(A_3)^{N_3'}) \\ N \cup N_3' \cup N_1' = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \mathrm{Tr}_{\mathrm{UN}}(A_3)^{N_3'} \cup \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_1'}) \end{cases}$$

which still correspond to the unique stable models of $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1 \cup A_2)$, $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_2 \cup A_3)$, and $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_3 \cup A_1)$, respectively.

We proceed by reducing the first equation in (4.6) using Lemma 4.2. Note that $\mathrm{Tr}_{\mathrm{UN}}(A_1) \cup \mathrm{Tr}_{\mathrm{UN}}(A_2) = \mathrm{Tr}_{\mathrm{UN}}(A_1 \cup A_2)$ by the modularity of $\mathrm{Tr}_{\mathrm{UN}}$ and $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(A_1 \cup A_2)) = \{\mathsf{a}, \mathsf{b}\} \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{UN}}(B))$. Thus we obtain $N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \{\mathsf{a} \leftarrow;\ \mathsf{b} \leftarrow\})$ by Lemma 4.2. Recall that N contains c in addition to a and b. Let $\mathrm{Tr}_{\mathrm{UN}}(B)_0^N$ and $\mathrm{Tr}_{\mathrm{UN}}(B)_1^N$ denote the disjoint sets of atomic and strictly unary rules of $\mathrm{Tr}_{\mathrm{UN}}(B)^N$, respectively. It follows by Lemma 4.6 that there is an atomic rule $\mathsf{d} \leftarrow$ in $\mathrm{Tr}_{\mathrm{UN}}(B)_0^N \cup \{\mathsf{a} \leftarrow;\ \mathsf{b} \leftarrow\}$ such that $\mathsf{c} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)_1^N \cup \{\mathsf{d} \leftarrow\})$. As $\mathrm{LM}(\cdot)$ is a monotonic operator, we obtain two cases: $\mathsf{c} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \{\mathsf{a} \leftarrow\})$ *or* $\mathsf{c} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \{\mathsf{b} \leftarrow\})$.

In the first case, we obtain $\mathsf{c} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_1'})$ by the monotonicity of the operator $\mathrm{LM}(\cdot)$ again. Recall that $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(A_3)) = \mathrm{Hb}_{\mathrm{v}}(A_3) = \{\mathsf{c}\}$ by the definition of $\mathrm{Tr}_{\mathrm{UN}}$. As a result of applying Lemma 4.2 to the third equation in (4.7), $\mathrm{Tr}_{\mathrm{UN}}(A_3)^{N_3'}$ is reduced to $\{\mathsf{c} \leftarrow\}$, so that $N \cup N_1' = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_1'} \cup \{\mathsf{c} \leftarrow\})$. Since c belongs to $\mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_1'})$ this simplifies to $N \cup N_1' = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B)^N \cup \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_1'})$. Because $N \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{UN}}(B))$, $N_1' \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{UN}}(A_1))$, and N and N_1' coincide on the atoms in $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(B)) \cap \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{UN}}(A_1)) = \{\mathsf{a}\}$, we get $\mathrm{Tr}_{\mathrm{UN}}(B)^N = \mathrm{Tr}_{\mathrm{UN}}(B)^{N \cup N_1'}$ and $\mathrm{Tr}_{\mathrm{UN}}(A_1)^{N_1'} = \mathrm{Tr}_{\mathrm{UN}}(A_1)^{N \cup N_1'}$. Then the modularity of $\mathrm{Tr}_{\mathrm{UN}}$ implies $N \cup N_1' = \mathrm{LM}(\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1)^{N \cup N_1'})$. Thus $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_1)$ possesses a stable model $N \cup N_1'$ containing $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$. A contradiction, since $B \cup A_1$ has a unique stable model $\{\mathsf{a}\}$.

In the second case, we can analogously construct a stable model $N \cup N_2'$ for $\mathrm{Tr}_{\mathrm{UN}}(B \cup A_2)$ using the second equation in (4.7). Again, this is a contradiction, as $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\} \subseteq N \cup N_2'$ and $\{\mathsf{b}\}$ is the unique stable model of $B \cup A_2$. □

Corollary 4.19 *$\mathcal{B} \not\leq_{\mathrm{PFM}} \mathcal{U}$, $\mathcal{U} <_{\mathrm{PFM}} \mathcal{B}$, and $\mathcal{U} <_{\mathrm{PFM}} \mathcal{P}$.*

It remains to explore the strictness of the relationship $\mathcal{A} \leq_{\mathrm{PFM}} \mathcal{U}$. At this point, it is worth demonstrating a particular translation technique [40, Proof of Theorem 3.10], which suitably exploits new atoms and negative literals

and thus serves as a potential candidate for a faithful and modular translation function from $\mathcal{U}$ to $\mathcal{A}$.

Example 4.20 Consider normal logic programs $P_1 = \{\mathsf{a} \leftarrow \mathsf{b}\}$ and $P_2 = \{\mathsf{b} \leftarrow \mathsf{c}\}$ and a translation of $P_1 \cup P_2$ into an atomic normal logic program

$$\begin{array}{rcl} \mathrm{Tr}_{\mathrm{AT}}(P_1 \cup P_2) & = & \mathrm{Tr}_{\mathrm{AT}}(P_1) \cup \mathrm{Tr}_{\mathrm{AT}}(P_2) \\ & = & \{\mathsf{a} \leftarrow {\sim}\overline{\mathsf{b}};\ \overline{\mathsf{b}} \leftarrow {\sim}\mathsf{b}\} \cup \{\mathsf{b} \leftarrow {\sim}\overline{\mathsf{c}};\ \overline{\mathsf{c}} \leftarrow {\sim}\mathsf{c}\} \end{array}$$

where the intuitive reading of the new atoms $\overline{\mathsf{b}}$ and $\overline{\mathsf{c}}$ is that b and c are false, respectively. The translation tries to capture the rules of P using a kind of double negation. In particular, the rules $\overline{\mathsf{b}} \leftarrow {\sim}\mathsf{b}$ and $\overline{\mathsf{c}} \leftarrow {\sim}\mathsf{c}$ can be understood to encode the standard closed world assumption [38]: b and c are false by default. The programs $P_1 \cup P_2$ and $\mathrm{Tr}_{\mathrm{AT}}(P_1 \cup P_2)$ behave as follows.

A	Stable models of $P_1 \cup P_2 \cup A$	Stable models of $\mathrm{Tr}_{\mathrm{AT}}(P_1 \cup P_2 \cup A)$
$\emptyset$	$\emptyset$	$\{\overline{\mathsf{b}}, \overline{\mathsf{c}}\}$
$\{\mathsf{a} \leftarrow\}$	$\{\mathsf{a}\}$	$\{\mathsf{a}, \overline{\mathsf{b}}, \overline{\mathsf{c}}\}$
$\{\mathsf{b} \leftarrow\}$	$\{\mathsf{a}, \mathsf{b}\}$	$\{\mathsf{a}, \mathsf{b}, \overline{\mathsf{c}}\}$
$\{\mathsf{c} \leftarrow\}$	$\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$	$\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$

By the preceding analysis, the translation $\mathrm{Tr}_{\mathrm{AT}}(P_1 \cup P_2)$ seems to capture the essentials of $P_1 \cup P_2$ in a modular and faithful manner. However, severe problems arise with programs containing an inferential loop that lets one to infer a from a, for instance. The simplest possible example of this kind is $P = \{\mathsf{a} \leftarrow \mathsf{a}\}$ having a minimal model $\mathrm{LM}(P) = \emptyset$. Unfortunately, the translation $\mathrm{Tr}_{\mathrm{AT}}(P) = \{\mathsf{a} \leftarrow {\sim}\overline{\mathsf{a}};\ \overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a}\}$ has two stable models $\{\overline{\mathsf{a}}\}$ and $\{\mathsf{a}\}$. The former model is what we would expect on the basis of our example, but the latter is a spurious stable model — dashing our hopes for $\mathrm{Tr}_{\mathrm{AT}}$ being faithful and modular in general. It is proved in the following theorem that the problems with $\mathrm{Tr}_{\mathrm{AT}}$ cannot be settled.

Theorem 4.21 $\mathcal{U} \not\leq_{\mathrm{FM}} \mathcal{A}$.

PROOF. Suppose there is a faithful and modular translation function $\mathrm{Tr}_{\mathrm{AT}}$ from $\mathcal{U}$ to $\mathcal{A}$. In the sequel, we analyze two unary normal programs $U_1 = \{\mathsf{a} \leftarrow \mathsf{b}\}$ and $U_2 = \{\mathsf{b} \leftarrow \mathsf{a}\}$, their combinations with atomic normal programs $A_1 = \{\mathsf{b} \leftarrow\}$ and $A_2 = \{\mathsf{a} \leftarrow\}$, and their translations under $\mathrm{Tr}_{\mathrm{AT}}$. To check the module conditions, we note that $\mathrm{Hb}_{\mathrm{v}}(U_1) = \mathrm{Hb}(U_1) = \{\mathsf{a}, \mathsf{b}\} = \mathrm{Hb}(U_2) = \mathrm{Hb}_{\mathrm{v}}(U_1)$, $\mathrm{Hb}_{\mathrm{v}}(A_1) = \mathrm{Hb}(A_1) = \{\mathsf{b}\}$, and $\mathrm{Hb}_{\mathrm{v}}(A_2) = \mathrm{Hb}(A_2) = \{\mathsf{a}\}$. Because there are no hidden atoms and the rules of the four programs are distinct, the module conditions are trivially satisfied by U_1 and A_1, by U_2 and A_2, by $U_1 \cup A_1$ and $U_2 \cup A_2$, and by $U_1 \cup U_2$ and $A_1 \cup A_2$.

The program $U_1 \cup A_1$ has a unique stable model $M_1 = \mathrm{LM}(U_1 \cup A_1) = \{\mathsf{a}, \mathsf{b}\}$. The translation $\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_1) = \mathrm{Tr}_{\mathrm{AT}}(U_1) \cup \mathrm{Tr}_{\mathrm{AT}}(A_1)$ and the modularity of $\mathrm{Tr}_{\mathrm{AT}}$ implies $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_1)) = \mathrm{Hb}_{\mathrm{v}}(U_1 \cup A_1) = \{\mathsf{a}, \mathsf{b}\}$. Since $\mathrm{Tr}_{\mathrm{AT}}$ is also faithful, the translation $\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_1)$ has a unique stable model $N_1 = \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_1)^{N_1}) = \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U_1)^{N_1} \cup \mathrm{Tr}_{\mathrm{AT}}(A_1)^{N_1})$ such

that $\{\mathsf{a},\mathsf{b}\} \subseteq N_1$. Then it holds by symmetry that $M_2 = \mathrm{LM}(U_2 \cup A_2) = \{\mathsf{a},\mathsf{b}\}$ is the unique stable model of $U_2 \cup A_2$ and $N_2 = \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U_2 \cup A_2)^{N_2})$ is the unique stable model of the translation $\mathrm{Tr}_{\mathrm{AT}}(U_2 \cup A_2) = \mathrm{Tr}_{\mathrm{AT}}(U_2) \cup \mathrm{Tr}_{\mathrm{AT}}(A_2)$, for which $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_2 \cup A_2)) = \mathrm{Hb}_{\mathrm{v}}(U_2 \cup A_2) = \{\mathsf{a},\mathsf{b}\}$ holds, so that $\{\mathsf{a},\mathsf{b}\} \subseteq N_2$.

Recall that $\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_1) = \mathrm{Tr}_{\mathrm{AT}}(U_1) \cup \mathrm{Tr}_{\mathrm{AT}}(A_1)$ is an atomic program and $\mathsf{a} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U_1)^{N_1} \cup \mathrm{Tr}_{\mathrm{AT}}(A_1)^{N_1})$. It follows by Lemma 4.9 that $\mathsf{a} \leftarrow$ must belong to the reduct. Since $\mathsf{a} \notin \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(A_1))$ and $\mathsf{a} \in \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_1))$ by the faithfulness of $\mathrm{Tr}_{\mathrm{AT}}$, and the translations $\mathrm{Tr}_{\mathrm{AT}}(U_1)$ and $\mathrm{Tr}_{\mathrm{AT}}(A_1)$ satisfy module conditions by the modularity of $\mathrm{Tr}_{\mathrm{AT}}$, we have $\mathsf{a} \notin \mathrm{Hb}(\mathrm{Tr}_{\mathrm{AT}}(A_1))$. Thus $\mathsf{a} \leftarrow$ cannot belong to $\mathrm{Tr}_{\mathrm{AT}}(A_1)^{N_1}$. So it must belong to $\mathrm{Tr}_{\mathrm{AT}}(U_1)^{N_1}$. It follows by symmetry that $\mathsf{b} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{AT}}(U_2)^{N_2}$.

Because $U_1 \cup A_1$ and $U_2 \cup A_2$ satisfy the module conditions, and N_1 and N_2 coincide up to the atoms in $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_2)) \cap \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_2 \cup A_2)) = \{\mathsf{a},\mathsf{b}\}$, we know by Lemma 4.3 that $N_1 \cup N_2$ is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_1) \cup \mathrm{Tr}_{\mathrm{AT}}(U_2 \cup A_2)$ which equals to $\mathrm{Tr}_{\mathrm{AT}}(U_1) \cup \mathrm{Tr}_{\mathrm{AT}}(A_1) \cup \mathrm{Tr}_{\mathrm{AT}}(U_2) \cup \mathrm{Tr}_{\mathrm{AT}}(A_2) = \mathrm{Tr}_{\mathrm{AT}}(U_1 \cup U_2) \cup \mathrm{Tr}_{\mathrm{AT}}(A_1 \cup A_2)$ by the modularity of $\mathrm{Tr}_{\mathrm{AT}}$. Moreover, the reduct $(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_1) \cup \mathrm{Tr}_{\mathrm{AT}}(U_2 \cup A_2))^{N_1 \cup N_2}$ is the union of $\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup A_1)^{N_1}$ and $\mathrm{Tr}_{\mathrm{AT}}(U_2 \cup A_2)^{N_2}$, i.e. $\mathrm{Tr}_{\mathrm{AT}}(U_1)^{N_1} \cup \mathrm{Tr}_{\mathrm{AT}}(U_2)^{N_2} \cup \mathrm{Tr}_{\mathrm{AT}}(A_1)^{N_1} \cup \mathrm{Tr}_{\mathrm{AT}}(A_2)^{N_2}$.

Since $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(A_1)) = \{\mathsf{b}\}$ and $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(A_2)) = \{\mathsf{a}\}$ are contained in $\mathrm{Hb}(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup U_2))$, Lemma 4.2 implies that $N = (N_1 \cup N_2) \cap \mathrm{Hb}(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup U_2))$ equals to $\mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U_1)^{N_1} \cup \mathrm{Tr}_{\mathrm{AT}}(U_2)^{N_2} \cup \{\mathsf{a} \leftarrow;\ \mathsf{b} \leftarrow\})$. Since $\mathsf{a} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{AT}}(U_1)^{N_1}$ and $\mathsf{b} \leftarrow$ to $\mathrm{Tr}_{\mathrm{AT}}(U_2)^{N_2}$, we can establish that N equals to $\mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U_1)^{N_1} \cup \mathrm{Tr}_{\mathrm{AT}}(U_2)^{N_2})$. Moreover, the equality $\mathrm{Tr}_{\mathrm{AT}}(U_1)^{N_1} \cup \mathrm{Tr}_{\mathrm{AT}}(U_2)^{N_2} = \mathrm{Tr}_{\mathrm{AT}}(U_1 \cup U_2)^{N}$ follows by the definition of N and the fact that N_1 and N_2 coincide on the atoms in $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_1)) = \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_2)) = \{\mathsf{a},\mathsf{b}\}$. Thus $N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup U_2)^{N})$ is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup U_2)$.

Since $\mathrm{Hb}_{\mathrm{v}}(U_1 \cup U_2) = \mathrm{Hb}(U_1 \cup U_2) = \{\mathsf{a},\mathsf{b}\}$ and $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup U_2)) = \mathrm{Hb}_{\mathrm{v}}(U_1 \cup U_2)$ by the faithfulness of $\mathrm{Tr}_{\mathrm{AT}}$, and $N \cap \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(U_1 \cup U_2)) = \{\mathsf{a},\mathsf{b}\}$, the faithfulness of $\mathrm{Tr}_{\mathrm{AT}}$ implies that $U_1 \cup U_2$ has a stable model $M = \{\mathsf{a},\mathsf{b}\}$. A contradiction, as $\emptyset$ is the unique stable model of $U_1 \cup U_2$. □

Corollary 4.22 *$\mathcal{U} \not\leq_{\mathrm{PFM}} \mathcal{A}$, $\mathcal{A} <_{\mathrm{PFM}} \mathcal{U}$, $\mathcal{A} <_{\mathrm{PFM}} \mathcal{B}$, and $\mathcal{A} <_{\mathrm{PFM}} \mathcal{P}$.*

In order to make our view complete, let us yet address the relationship between positive programs and normal programs. Recall that by Proposition 3.18, $\mathcal{C}^+ \leq_{\mathrm{PFM}} \mathcal{C}$ holds for any of the classes $\mathcal{C}$ under consideration.

Theorem 4.23 *For any $\mathcal{C} \in \{\mathcal{A},\mathcal{U},\mathcal{B},\mathcal{P}\}$, $\mathcal{C} \not\leq_{\mathrm{F}} \mathcal{C}^+$.*

PROOF. Let us assume that there is a faithful translation function Tr from $\mathcal{C}$ to $\mathcal{C}^+$. Consider a logic program $P = \{\mathsf{a} \leftarrow \sim\mathsf{a}\}$ which serves as a representative of the class $\mathcal{C}$. Let Q be the translation $\mathrm{Tr}(P)$ in $\mathcal{C}^+$. Now P has no stable models, but the translation Q has a unique stable model $\mathrm{LM}(Q)$, which contradicts the faithfulness of Tr. □

Corollary 4.24 *For any $\mathcal{C} \in \{\mathcal{A},\mathcal{U},\mathcal{B},\mathcal{P}\}$, $\mathcal{C} \not\leq_{\mathrm{PFM}} \mathcal{C}^+$ and $\mathcal{C}^+ <_{\mathrm{PFM}} \mathcal{C}$.*

The resulting hierarchy of classes of logic programs is illustrated in Figure 1.

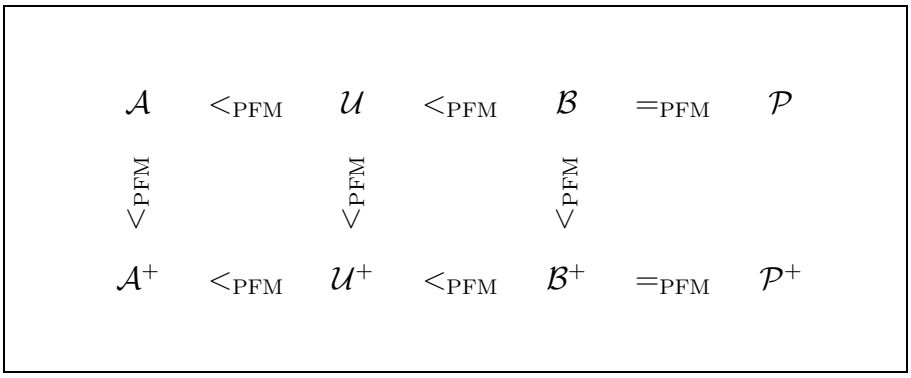

Figure 1: Expressive Power Hierarchy based on polynomial, faithful and modular (PFM) translation functions

4.4 Comparison with Propositional Logic

It is worthwhile to relate our framework with propositional logic. It is assumed that a propositional theory S is given as a set of clauses of the form (2.3).[5] The fundamental *satisfiability problem* (SAT) is about checking if a given set of clauses S has a model in the sense of Definition 2.6. However, in this report, we are interested in all models of S rather than checking the existence of a model. This is because it is assumed that the models correspond to the solutions of the problem formalized as a set of clauses S. It is possible to capture the models of a set of clauses S with the stable models of a translation of S into a normal program. In fact, we can do this using only atomic rules. The basic idea is as follows. The rules $\mathsf{a} \leftarrow {\sim}\overline{\mathsf{a}}$ and $\overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a}$ are needed to select the truth value of each atom $\mathsf{a} \in \mathrm{Hb}(S)$. Here $\overline{\mathsf{a}} \notin \mathrm{Hb}(S)$ is a new atom meaning that a is false (c.f. Example 4.20). Given these rules, we obtain all model candidates for S as the stable models of the rules. Yet we have to ensure that all clauses of the form (2.3) are satisfied — this is accomplished by introducing a new atom $\mathsf{f} \notin \mathrm{Hb}(S)$ and an atomic rule

$$\mathsf{f} \leftarrow {\sim}\mathsf{f}, {\sim}\mathsf{a}_1, \ldots, {\sim}\mathsf{a}_n, {\sim}\overline{\mathsf{b}}_1, \ldots, {\sim}\overline{\mathsf{b}}_m \tag{4.8}$$

for each clause (2.3) in S. These kinds of rules exclude model candidates in which some of the clauses is false. If the full syntax of normal programs is assumed, then it would be more intuitive to use a rule of the form $\mathsf{f} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_m, {\sim}\mathsf{f}, {\sim}\mathsf{a}_1, \ldots, {\sim}\mathsf{a}_n$ which is not atomic, but "double negation" is needed in order to make the rule atomic. Yet another technique is given in [36]: a new atom c is introduced for each clause (2.3) which is translated into

$$\mathsf{c} \leftarrow \mathsf{a}_1;\ \ldots;\ \mathsf{c} \leftarrow \mathsf{a}_n;\ \mathsf{c} \leftarrow \overline{\mathsf{b}_1};\ \ldots;\ \mathsf{c} \leftarrow \overline{\mathsf{b}_m}.$$

Then (4.8) can be replaced by $\mathsf{f} \leftarrow {\sim}\mathsf{f}, {\sim}\mathsf{c}$. However, to meet the module conditions from Definition 3.15, we have to localize the choice of truth values. For this reason we translate a clause c of the form (2.3) into a set of

[5] Any propositional theory can be transformed into clausal form and the transformation is polynomial if new atoms can be introduced. Without new atoms it is difficult to transform formulas such as $(\mathsf{a}_1 \wedge \neg\mathsf{a}_1) \vee \cdots \vee (\mathsf{a}_n \wedge \neg\mathsf{a}_n)$ into clausal form.

rules

$$\begin{aligned}\mathrm{Tr}_{\mathrm{LP}}(c) = {} & \{\mathsf{f}^c \leftarrow {\sim}\mathsf{f}^c, {\sim}\mathsf{a}_1, \ldots, {\sim}\mathsf{a}_n, {\sim}\mathsf{b}_1^c, \ldots, {\sim}\mathsf{b}_m^c\} \cup \\ & \{\mathsf{a}_i \leftarrow {\sim}\mathsf{a}_i^c;\ \mathsf{a}_i^c \leftarrow {\sim}\mathsf{a}_i \mid 0 < i \leq n\} \cup \\ & \{\mathsf{b}_i \leftarrow {\sim}\mathsf{b}_i^c;\ \mathsf{b}_i^c \leftarrow {\sim}\mathsf{b}_i \mid 0 < i \leq n\}\end{aligned} \tag{4.9}$$

where f^c, $\mathsf{a}_1^c, \ldots, \mathsf{a}_n^c$, and $\mathsf{b}_1^c, \ldots, \mathsf{b}_m^c$ are new atoms that are unique to c. This implies that the choice of the truth value of an atom a is shared by the rules in which the atom appears. However, these choices are synchronized, as a is shared among the rules, and this is how a is assigned a unique truth value.

Definition 4.25 *A set of clauses S is translated into*

$$\mathrm{Tr}_{\mathrm{LP}}(S) = \bigcup\{\mathrm{Tr}_{\mathrm{LP}}(c) \mid c \in S\} \cup \{\mathsf{a} \leftarrow {\sim}\overline{\mathsf{a}};\ \overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a} \mid \mathsf{a} \in \mathrm{Hb}_{\mathrm{a}}(S)\}$$

with $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{LP}}(S)) = \emptyset$ *and* $\mathrm{Hb}(\mathrm{Tr}_{\mathrm{LP}}(S)) =$

$$\mathrm{Hb}(S) \cup \{\mathsf{f}^c \mid c \in S\} \cup \{\mathsf{a}^c \mid c \in S \text{ and } \mathsf{a} \text{ appears in } c\} \cup \{\overline{\mathsf{a}} \mid \mathsf{a} \in \mathrm{Hb}_{\mathrm{a}}(S)\}.$$

The visible part $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{LP}}(S)) = \mathrm{Hb}_{\mathrm{v}}(S)$.

A particular feature of the translation $\mathrm{Tr}_{\mathrm{LP}}(S)$ is that all atoms of $\mathrm{Hb}(S)$ actually appear in the rules of $\mathrm{Tr}_{\mathrm{LP}}(S)$ and thus $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{LP}}(S))$ can be left empty. The rules associated with the atoms in $\mathrm{Hb}_{\mathrm{a}}(S)$ are necessary in order to capture the classical models of S properly, since $\mathrm{Hb}(S)$ may contain atoms that do not appear in the clauses of S; and according to Definition 2.6 classical models of S are subsets of $\mathrm{Hb}(S)$. Given a set of clauses S, an interpretation $I \subseteq \mathrm{Hb}(S)$, and a clause $c \in S$, we define the set of *complementary atoms* $\mathrm{CA}(c, M)$ which is to contain a^c whenever a appears in c and $\mathsf{a} \notin M$. For the set S as whole, we let

$$\mathrm{CA}(S, M) = \bigcup\{\mathrm{CA}(c, M) \mid c \in S\} \cup \{\overline{\mathsf{a}} \mid \mathsf{a} \in \mathrm{Hb}_{\mathrm{a}}(S) - M\} \tag{4.10}$$

which takes also the additional atoms from $\mathrm{Hb}_{\mathrm{a}}(S)$ properly into account. We are now ready to address the correctness of the translation function $\mathrm{Tr}_{\mathrm{LP}}$.

Proposition 4.26 *Let S be a set of clauses. If $M \subseteq \mathrm{Hb}(S)$ is a classical model of S, then $N = M \cup \mathrm{CA}(S, M)$ is a stable model of $\mathrm{Tr}_{\mathrm{LP}}(S)$ such that $N \cap \mathrm{Hb}(S) = M$.*

PROOF. Let M be a (classical) model of S and define N as above. The reduct $\mathrm{Tr}_{\mathrm{LP}}(S)^N$ contains (i) $\mathsf{a} \leftarrow$ for each $\mathsf{a} \in M$, (ii) $\mathsf{a}^c \leftarrow$ for each occurrence of an atom $\mathsf{a} \in \mathrm{Hb}(S) - M$ in a clause $c \in S$, and (iii) $\overline{\mathsf{a}} \leftarrow$ for each $\mathsf{a} \in \mathrm{Hb}_{\mathrm{a}}(S) - M$. In particular, the rule $\mathsf{f}^c \leftarrow$ is not included for any clause $c \in S$, because $N \cap \{\mathsf{f}^c, \mathsf{a}_1, \ldots, \mathsf{a}_n, \mathsf{b}_1^c, \ldots, \mathsf{b}_m^c\} = \emptyset$ would imply that (2.3) is not satisfied in M. Thus $N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{LP}}(S)^N)$ by Lemma 4.9 and N is a stable model of $\mathrm{Tr}_{\mathrm{LP}}(S)$. Since $\mathrm{CA}(S, M)$ contains only atoms that are new relative to $\mathrm{Hb}(S)$, we have $M = N \cap \mathrm{Hb}(S)$ by the definition of N. □

Proposition 4.27 *Let S be a set of clauses. If $N \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{LP}}(S))$ is a stable model of $\mathrm{Tr}_{\mathrm{LP}}(S)$, then $M = N \cap \mathrm{Hb}(S) \models S$ and $N = M \cup \mathrm{CA}(S, M)$.*

PROOF. Let $N \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{LP}}(S))$ be a stable model of $\mathrm{Tr}_{\mathrm{LP}}(S)$ and define M as above. Let us first assume that $\mathsf{f}^c \in N$ for some $c \in S$. It follows by Lemma 4.9 that $\mathsf{f}^c \leftarrow$ must belong to $\mathrm{Tr}_{\mathrm{LP}}(S)^N$. By the first rule in (4.9) this means that $\mathsf{f}^c \notin N$, a contradiction. Then consider any a^c such that $\mathsf{a} \in \mathrm{Hb}(S)$ appears in $c \in S$. Now $\mathsf{a}^c \in N \iff \mathsf{a}^c \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{LP}}(S)^N$ by Lemma 4.9 $\iff \mathsf{a} \notin N$ by (4.9) $\iff \mathsf{a} \notin M \iff \mathsf{a}^c \in \mathrm{CA}(S, M)$. Finally, let a be any atom from $\mathrm{Hb}_{\mathrm{a}}(S)$. Now $\overline{\mathsf{a}} \in N \iff \overline{\mathsf{a}} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{LP}}(S)^N$ by Lemma 4.9 $\iff \mathsf{a} \notin N$ by the rule $\overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a}$ included in $\mathrm{Tr}_{\mathrm{LP}}(S) \iff \mathsf{a} \notin M \iff \overline{\mathsf{a}} \in \mathrm{CA}(S, M)$. Recalling $\mathrm{Hb}(\mathrm{Tr}_{\mathrm{LP}}(S))$ from Definition 4.25, we conclude that $N = M \cup \mathrm{CA}(S, M)$ holds.

Suppose that $M \not\models S$, i.e. there is a clause $c \in S$ of the form (2.3) not satisfied in M. It follows that $M \cap \{\mathsf{a}_1, \ldots, \mathsf{a}_n\} = \emptyset$ and $\{\mathsf{b}_1, \ldots, \mathsf{b}_m\} \subseteq M$. The same relations hold w.r.t. N by the definition of M. Thus the rules $\mathsf{b}_1^c \leftarrow, \ldots, \mathsf{b}_m^c \leftarrow$ are not included in the reduct $\mathrm{Tr}_{\mathrm{LP}}(S)^N$ so that $N \cap \{\mathsf{a}_1, \ldots, \mathsf{a}_n, \mathsf{b}_1^c, \ldots, \mathsf{b}_m^c\} = \emptyset$ by Lemma 4.9. As shown above, we know that $\mathsf{f}^c \notin N$. Then the rule $\mathsf{f}^c \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{LP}}(S)^N$ so that $\mathsf{f}^c \in N$, contradiction. Hence M is a model of S. □

Theorem 4.28 $\mathcal{SC} \leq_{\mathrm{PFM}} \mathcal{A}$.

PROOF. The translation function $\mathrm{Tr}_{\mathrm{LP}}$ transforms a clause (2.3) consisting of $2n + 3m + 2$ symbols into a set of rules (4.9) with at most $3n + 3m + 5 + n \times (5+5) + m \times (5+5) = 13n + 13m + 5$ symbols. The translation of each atom $\mathsf{a} \in \mathrm{Hb}_{\mathrm{a}}(S)$ consists of 10 symbols. Moreover, definition 4.25 suggests that $\mathrm{Tr}_{\mathrm{LP}}(S)$ can be produced on a clause-by-clause basis in a linear number of steps. Thus we conclude that $\mathrm{Tr}_{\mathrm{LP}}$ is linear as well as polynomial.

To establish modularity, let S and T be two sets of clauses such that the module conditions M1–M4 from Definition 3.15 are satisfied. Definition 4.25 implies that $\mathrm{Tr}_{\mathrm{LP}}(S \cup T)$ equals to the union of $\mathrm{Tr}_{\mathrm{LP}}(S)$ and $\mathrm{Tr}_{\mathrm{LP}}(T)$. The module condition M1 is satisfied by the translations $\mathrm{Tr}_{\mathrm{LP}}(S)$ and $\mathrm{Tr}_{\mathrm{LP}}(T)$, as $S \cap T = \emptyset$ by M1, $\mathrm{Hb}_{\mathrm{a}}(S) \cap \mathrm{Hb}_{\mathrm{a}}(T) = \emptyset$ by M2, and the introduction of new atoms guarantee that the sets of rules (4.9) and $\{\mathsf{a} \leftarrow {\sim}\overline{\mathsf{a}};\ \overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a} \mid \mathsf{a} \in \mathrm{Hb}_{\mathrm{a}}(S)\}$ that constitute $\mathrm{Tr}_{\mathrm{LP}}(S)$ remain distinct from those of $\mathrm{Tr}_{\mathrm{LP}}(T)$. The module condition M2 is trivially satisfied, since $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{LP}}(S)) = \mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{LP}}(T)) = \emptyset$ by definition. In addition, M3 and M4 remain valid, as $\mathrm{Tr}_{\mathrm{LP}}$ adds new atoms that remain local to the translations of S and T.

It remains to show $\mathrm{Tr}_{\mathrm{LP}}$ faithful. First, Definition 4.25 implies for any set of clauses that $\mathrm{Hb}(S) \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{LP}}(S))$ and $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{LP}}(S)) = \mathrm{Hb}_{\mathrm{v}}(S)$. In Proposition 4.26, we show that $\mathrm{Ext}_{\mathrm{LP}} : \mathrm{CM}(S) \to \mathrm{SM}(\mathrm{Tr}_{\mathrm{LP}}(S))$ is an extension function which maps $M \in \mathrm{CM}(S)$ to $N = \mathrm{Ext}_{\mathrm{LP}}(M) = M \cup \mathrm{CA}(S, M)$ included in $\mathrm{SM}(\mathrm{Tr}_{\mathrm{LP}}(S))$ such that $M = N \cap \mathrm{Hb}(S)$. Moreover, we know by Proposition 4.27 that if $N \in \mathrm{SM}(\mathrm{Tr}_{\mathrm{LP}}(S))$, then $M = N \cap \mathrm{Hb}(P) \in \mathrm{CM}(S)$ and $N = \mathrm{Ext}_{\mathrm{LP}}(M)$. The rest follows by Theorem 3.19. □

On the other hand, it is impossible to translate an atomic normal program P into a set of clauses in a faithful and modular way. This result has been established by Niemelä [36, Proposition 4.3] for normal programs in general, but different notions of faithfulness and modularity are employed in Niemelä's proof.

Theorem 4.29 $\mathcal{A} \not\leq_{\mathrm{FM}} \mathcal{SC}$.

PROOF. Let us assume that there exists a faithful and modular translation function $\mathrm{Tr} : \mathcal{A} \to \mathcal{SC}$. Then consider atomic normal logic programs $P_1 = \{\mathsf{a} \leftarrow {\sim}\mathsf{a}\}$ and $P_2 = \{\mathsf{a} \leftarrow\}$ with $\mathrm{Hb}_{\mathrm{v}}(P_1) = \mathrm{Hb}(P_1) = \mathrm{Hb}_{\mathrm{v}}(P_2) = \mathrm{Hb}(P_2) = \{\mathsf{a}\}$. It is clear that P_1 and P_2 satisfy the module conditions from Definition 3.15. The program P_1 has no stable models while P_2 has a unique stable model $M = \{\mathsf{a}\}$. Since Tr is faithful, the translation $\mathrm{Tr}(P_1)$ must be propositionally inconsistent. By the modularity of Tr, the translation $\mathrm{Tr}(P_1 \cup P_2) = \mathrm{Tr}(P_1) \cup \mathrm{Tr}(P_2)$ which is also propositionally inconsistent, i.e. has no models. But this contradicts the faithfulness of Tr, since M is also the unique stable model of $P_1 \cup P_2$. Hence there is no such Tr. □

Corollary 4.30 *$\mathcal{A} \not\leq_{\mathrm{PFM}} \mathcal{SC}$, $\mathcal{SC} <_{\mathrm{PFM}} \mathcal{A}$, $\mathcal{SC} <_{\mathrm{PFM}} \mathcal{U}$, $\mathcal{SC} <_{\mathrm{PFM}} \mathcal{B}$, and $\mathcal{SC} <_{\mathrm{PFM}} \mathcal{P}$.*

Theorem 4.31 *$\mathcal{SC} \not\leq_{\mathrm{F}} \mathcal{C}^+$ holds for any $\mathcal{C} \in \{\mathcal{A}, \mathcal{U}, \mathcal{B}, \mathcal{P}\}$.*

PROOF. The set of clauses $S = \{\{\mathsf{a}, \neg\mathsf{a}\}\}$ has two classical models $M_1 = \emptyset$ and $M_2 = \{\mathsf{a}\}$ which cannot be faithfully captured by a positive program $P = \mathrm{Tr}(S)$ possessing a unique stable model $\mathrm{LM}(P)$. □

Theorem 4.32 *$\mathcal{C}^+ \not\leq_{\mathrm{FM}} \mathcal{SC}$ holds for all classes $\mathcal{C} \in \{\mathcal{A}, \mathcal{U}, \mathcal{B}, \mathcal{P}\}$.*

PROOF. We begin with the class $\mathcal{U}$ by assuming the existence of a faithful and modular translation function $\mathrm{Tr}_{\mathrm{CL}}$ from $\mathcal{U}^+$ to $\mathcal{SC}$. To establish a counter-example, we use positive unary programs $P_1 = \{\mathsf{a} \leftarrow\}$ and $P_2 = \{\mathsf{a} \leftarrow \mathsf{a}\}$ which satisfy the module conditions, as $\mathrm{Hb}_{\mathrm{v}}(P_i) = \mathrm{Hb}(P_i) = \{\mathsf{a}\}$ for $i \in \{1, 2\}$. Now P_1 and P_2 have unique stable models $\mathrm{LM}(P_1) = \{\mathsf{a}\}$ and $\mathrm{LM}(P_2) = \emptyset$, respectively. It follows by the faithfulness of $\mathrm{Tr}_{\mathrm{CL}}$ that the unique classical models of $\mathrm{Tr}_{\mathrm{CL}}(P_1)$ and $\mathrm{Tr}_{\mathrm{CL}}(P_2)$ are N_1 and N_2 such that $M_1 = N_1 \cap \{\mathsf{a}\}$ and $M_2 = N_2 \cap \{\mathsf{a}\}$. It follows that $\mathrm{Tr}_{\mathrm{CL}}(P_1) \models \mathsf{a}$ and $\mathrm{Tr}_{\mathrm{CL}}(P_2) \models \neg\mathsf{a}$. On the other hand, we know by the modularity of $\mathrm{Tr}_{\mathrm{CL}}$ that $\mathrm{Tr}_{\mathrm{CL}}(P_1 \cup P_2) = \mathrm{Tr}_{\mathrm{CL}}(P_1) \cup \mathrm{Tr}_{\mathrm{CL}}(P_2)$. It follows that $\mathrm{Tr}_{\mathrm{CL}}(P_1 \cup P_2) \models \mathsf{a} \wedge \neg\mathsf{a}$, i.e. $\mathrm{Tr}_{\mathrm{CL}}(P_1 \cup P_2)$ has no models. However, the program $P_1 \cup P_2$ has a unique stable model $\mathrm{LM}(P_1 \cup P_2) = \{\mathsf{a}\}$ which contradicts the faithfulness of $\mathrm{Tr}_{\mathrm{CL}}$.

The programs P_1 and P_2 above are also representatives of the classes $\mathcal{B}^+$ and $\mathcal{P}^+$. In this way, the proof above covers classes $\mathcal{C} \in \{\mathcal{B}, \mathcal{P}\}$, too. The fate of $\mathcal{A}$ needs reconsideration as $P_2 \notin \mathcal{A}^+$. In fact, we may substitute $P_2 = \emptyset$ with $\mathrm{Hb}_{\mathrm{a}}(P_2) = \{\mathsf{a}\}$ for P_2 above to obtain a proof of $\mathcal{A}^+ \not\leq_{\mathrm{FM}} \mathcal{SC}$. □

By the preceding results, we may make precise the position of $\mathcal{SC}$ in the expressive power hierarchy illustrated in Figure 1. That is, $\mathcal{SC}$ is strictly less expressive than any of the classes $\mathcal{A}, \mathcal{U}, \mathcal{B}$, and $\mathcal{P}$, but incomparable with the respective classes of positive programs.

Corollary 4.33 *$\mathcal{C}^+ \neq_{\mathrm{PFM}} \mathcal{SC}$ holds for all classes $\mathcal{C} \in \{\mathcal{A}, \mathcal{U}, \mathcal{B}, \mathcal{P}\}$.*

5 YET ANOTHER CHARACTERIZATION OF STABILITY

In Section 6, we will study non-modular alternatives to translation functions that were shown not to exist in Section 4. As a preparation for one particular translation function, we will address an alternative semantics for normal programs, namely the one based on *supported models* [2]. It turns out in the sequel that we are able to *characterize* stable models as supported models possessing a level numbering # which extends the notion of level numbers, first introduced for positive programs in Section 2.2, to the case of normal programs.

Definition 5.1 *A classical model M of a normal program P is a supported model of P if and only if for every atom $\mathsf{a} \in M$ there is a rule $r \in P$ such that* $\mathrm{head}(r) = \mathsf{a}$ *and* $M \models \mathrm{body}(r)$.

Inspired by this notion, we define the set of *supporting rules* $\mathrm{SR}(P, I) = \{r \in P \mid I \models \mathrm{body}(r)\} \subseteq P$ for any normal program P and an interpretation $I \subseteq \mathrm{Hb}(P)$. Thus the intuition behind this set is that each rule $r \in \mathrm{SR}(P, I)$ provides a support for $\mathrm{head}(r)$.

Definition 5.2 *Let M be a supported model of a normal program P. A function # from $M \cup \mathrm{SR}(P, M)$ to $\mathbb{N}$ is a level numbering w.r.t. M iff*

$$\forall \mathsf{a} \in M : \#\mathsf{a} = \min\{\#r \mid r \in \mathrm{SR}(P, M) \text{ and } \mathsf{a} = \mathrm{head}(r)\} \tag{5.1}$$

and

$$\forall r \in \mathrm{SR}(P, M) : \quad \#r = \begin{cases} \max\{\#\mathsf{b} \mid \mathsf{b} \in \mathrm{body}^+(r)\} + 1, & \text{if } \mathrm{body}^+(r) \neq \emptyset. \\ 1, & \text{otherwise.} \end{cases} \tag{5.2}$$

It is important to realize that a level numbering need not exist for every supported model. This is demonstrated by the following example.

Example 5.3 Consider a logic program P consisting of two rules $r_1 = \mathsf{a} \leftarrow \mathsf{b}$ and $r_2 = \mathsf{b} \leftarrow \mathsf{a}$. There are two supported models of P: $M_1 = \emptyset$ and $M_2 = \{\mathsf{a}, \mathsf{b}\}$. The first model has a trivial level numbering with an empty domain, since $M_1 \cup \mathrm{SR}(P, M_1) = \emptyset$. For the second, the domain $M_2 \cup \mathrm{SR}(P, M_2) = M_2 \cup P$. The requirements in Definition 5.2 lead to four equations: $\#\mathsf{a} = \#r_1$, $\#r_1 = \#\mathsf{b} + 1$, $\#\mathsf{b} = \#r_2$, and $\#r_2 = \#\mathsf{a} + 1$. From these, we obtain $\#\mathsf{a} = \#\mathsf{a} + 2$. Hence there is no level numbering w.r.t. M_2.

Proposition 5.4 *Let M be a supported model of P. If there is a level numbering w.r.t. M, then the level numbering is unique.*

PROOF. Suppose that $\#_1$ and $\#_2$ are two level numberings w.r.t. M. It is proved by induction on $\#_1(x) > 0$ for $x \in M \cup \mathrm{SR}(P, M)$ that $\#_1(x) = \#_2(x)$.

Base case: $\#_1(x) = 1$. If $x = \mathsf{a}$ for an atom $\mathsf{a} \in M$, then $\#_1(\mathsf{a}) = 1$ $\iff$ there is a rule $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$ and $\mathrm{body}^+(r) = \emptyset$

$\iff \#_2(\mathsf{a}) = 1$. Similarly, if $x = r$ for for a rule $r \in \mathrm{SR}(P, M)$, then $\#_1(r) = 1 \iff \mathrm{body}^+(r) = \emptyset \iff \#_2(r) = 1$.

Induction step: $\#_1(x) = n > 1$. If $x = r$ for a rule $r \in \mathrm{SR}(P, M)$, then $\#_1(r) = \max\{\#_1(\mathsf{b}) \mid \mathsf{b} \in \mathrm{body}^+(r)\} + 1$. Note that $\mathrm{body}^+(r) \neq \emptyset$, as $\#_1(r) > 1$. Then consider any $\mathsf{b} \in \mathrm{body}^+(r)$. Since $\#_1(\mathsf{b}) < n$ we obtain $\#_2(\mathsf{b}) = \#_1(\mathsf{b})$ by the inductive hypothesis. Thus $\#_2(r) = \#_1(r)$ holds, too. If $x = \mathsf{a}$ for an atom $\mathsf{a} \in M$, then $\#_1(\mathsf{a}) = \#_1(r)$ for some rule $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$ yielding the minimum value for $\#_1(r)$. By the preceding analysis, we know that $\#_2(r) = \#_1(r)$ for this particular rule so that $\#_2(\mathsf{a}) \leq \#_1(\mathsf{a})$. Then suppose that $\#_2(\mathsf{a}) < \#_1(\mathsf{a})$, i.e. there is a rule $r' \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r') = \mathsf{a}$ and $\#_2(r') < \#_2(r)$. By the inductive hypothesis, $\#_1(r') = \#_2(r')$ which implies that $\#_1(r') < \#_1(\mathsf{a})$, a contradiction. Hence $\#_2(\mathsf{a}) = \#_1(\mathsf{a})$ is necessarily the case. □

The question is how one can determine level numberings in practice. In fact, the scheme introduced in Section 2.2 can be extended to cover rules as well.

Definition 5.5 *Let P be a positive program and $M = \mathrm{LM}(P)$. Let us define level numbers $\mathrm{lev}(\mathsf{a})$ for atoms $\mathsf{a} \in M$ as in Section 2.2. Given any rule $r \in P$ such that $\mathrm{body}^+(r) = \mathrm{body}(r) \subseteq M$, define the level number*

$$\mathrm{lev}(r) = \begin{cases} \max\{\mathrm{lev}(\mathsf{b}) \mid \mathsf{b} \in \mathrm{body}^+(r)\} + 1, & \textit{if } \mathrm{body}^+(r) \neq \emptyset. \\ 1, & \textit{otherwise.} \end{cases} \tag{5.3}$$

Assigning level numbers in this way is compatible with Definition 5.2.

Lemma 5.6 *Let P be a positive program, $M = \mathrm{LM}(P)$, and $\mathsf{a} \in M$.*

1. *For every $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$, $\mathrm{lev}(r) \geq \mathrm{lev}(\mathsf{a})$.*
2. *There is $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$ and $\mathrm{lev}(r) = \mathrm{lev}(\mathsf{a})$.*

PROOF. Recall that $\mathrm{lev}(\mathsf{a})$ is the least number $n > 0$ such that $\mathsf{a} \in \mathrm{T}_P \uparrow n$ but $\mathsf{a} \notin \mathrm{T}_P \uparrow n - 1$. Since $\mathsf{a} \in \mathrm{T}_P \uparrow n$, there is a rule $r \in P$ such that $\mathrm{head}(r) = \mathsf{a}$ and $\mathrm{body}^+(r) \subseteq \mathrm{T}_P \uparrow n - 1$ which is contained in $M = \mathrm{lfp}(\mathrm{T}_P)$. It follows that $r \in \mathrm{SR}(P, M)$ and $\mathrm{lev}(\mathsf{b}) < n$ for any $\mathsf{b} \in \mathrm{body}^+(r)$. Thus $\mathrm{lev}(r) \leq n$ holds by the definition of $\mathrm{lev}(r)$ in (5.3). To conclude, there is a rule $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$ and $\mathrm{lev}(r) \leq \mathrm{lev}(\mathsf{a})$.

1. Let $r' \in \mathrm{SR}(P, M)$ be any rule such that $\mathrm{head}(r') = \mathsf{a}$. Then suppose that $0 < \mathrm{lev}(r') < n$. Two cases arise on the basis of (5.3). (i) If $\mathrm{body}^+(r') = \mathrm{body}(r') = \emptyset$, then $\mathrm{lev}(r') = 1$ and $\mathsf{a} \in \mathrm{T}_P \uparrow 1$. On the other hand $\mathrm{lev}(r') < n$ implies $\mathsf{a} \notin \mathrm{T}_P \uparrow 1$, a contradiction. (ii) If $\mathrm{body}^+(r') = \mathrm{body}(r') \neq \emptyset$, we have $\mathrm{body}^+(r') \subseteq M$, as $r' \in \mathrm{SR}(P, M)$, and $\mathrm{lev}(\mathsf{b}) < \mathrm{lev}(r')$ for each $\mathsf{b} \in \mathrm{body}^+(r')$. It follows that $\mathrm{body}^+(r') \subseteq \mathrm{T}_P \uparrow m$ where m is the maximum value among $\{\mathrm{lev}(\mathsf{b}) \mid \mathsf{b} \in \mathrm{body}^+(r')\}$. Thus $\mathsf{a} \in \mathrm{T}_P \uparrow \mathrm{lev}(r')$. But this is a contradiction, as $\mathrm{lev}(r') \leq n - 1$ and $\mathsf{a} \notin \mathrm{T}_P \uparrow n - 1$. To conclude, $\mathrm{lev}(r') \geq \mathrm{lev}(\mathsf{a})$.
2. By the first item, we have $\mathrm{lev}(r) = \mathrm{lev}(\mathsf{a})$ for the rule r above. □

The characterization of stable models is then established as follows.

Proposition 5.7 *[33, Corollary 1] Any stable model $M \subseteq \mathrm{Hb}(P)$ of a normal logic program P is also a supported model of P.*

Theorem 5.8 *Let P be a normal program.*

1. *If M is a stable model of P, then M is a supported model of P and there exists a unique level numbering $\# : M \cup \mathrm{SR}(P, M) \to \mathbb{N}$ w.r.t. M defined as follows. (i) For $\mathsf{a} \in M$, let $\#\mathsf{a} = \mathrm{lev}(\mathsf{a})$. (ii) For $r \in \mathrm{SR}(P, M)$, let $\#r = \mathrm{lev}(r^+)$.*

2. *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.*

PROOF. (Item 1). Let M be a stable model of P. Then M is also a supported model of P by Proposition 5.7. Recall that each $r \in \mathrm{SR}(P, M)$ satisfies $M \models \mathrm{body}(r)$ which implies that $r^+ \in P^M$, $\mathrm{body}^+(r) \subseteq M$, and $\mathrm{head}(r) \in M$, as M is also a classical model of P. Let us now prove that the level numbers meet the requirements of Definition 5.2.

Consider any $\mathsf{a} \in M$. It should be established that $\#\mathsf{a}$ is the minimum among $\{\#r \mid r \in \mathrm{SR}(P, M) \text{ and } \mathrm{head}(r) = \mathsf{a}\}$. It is clear that this set is non-empty, as M is a supported model of P. Then consider any $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$. Now $\#r$ is defined as $\mathrm{lev}(r^+)$ given in (5.3). Since $r \in \mathrm{SR}(P, M)$, we obtain $r^+ \in \mathrm{SR}(P^M, M)$. Thus $\#r = \mathrm{lev}(r^+) \geq \mathrm{lev}(\mathsf{a})$ by the first claim of Lemma 5.6. By the second claim, there is a rule $r' \in \mathrm{SR}(P^M, M)$ such that $\mathrm{head}(r') = \mathsf{a}$ and $\mathrm{lev}(r') = \mathrm{lev}(\mathsf{a})$. Then there is a rule $r'' \in \mathrm{SR}(P, M)$ such that $r' = (r'')^+$, $\mathrm{head}(r'') = \mathsf{a}$ and $\#r'' = \mathrm{lev}(r'') = \mathrm{lev}(\mathsf{a})$. Thus $\#\mathsf{a} = \mathrm{lev}(\mathsf{a})$ is the minimum in question. Then consider any $r \in \mathrm{SR}(P, M)$. There are two possibilities. If $\mathrm{body}^+(r) = \emptyset$, we obtain $\mathrm{body}^+(r^+) = \emptyset$ so that $\#r = \mathrm{lev}(r^+) = 1$ by (5.3). This is in perfect harmony with Definition 5.2. On the other hand, if $\mathrm{body}^+(r) \neq \emptyset$, $\#r = \mathrm{lev}(r^+)$ is defined as $\max\{\mathrm{lev}(\mathsf{b}) \mid \mathsf{b} \in \mathrm{body}^+(r^+)\} + 1$. Since $\mathrm{body}^+(r^+) = \mathrm{body}^+(r)$ and $\#\mathsf{b} = \mathrm{lev}(\mathsf{b})$ for each $\mathsf{b} \in \mathrm{body}^+(r)$ by definition, $\#r = \mathrm{lev}(r^+)$ coincides with $\max\{\#\mathsf{b} \mid \mathsf{b} \in \mathrm{body}^+(r)\} + 1$ as insisted by Definition 5.2.

We may conclude that $\#$ is a level numbering w.r.t. M. Since M is a supported model of P, the uniqueness of $\#$ follows by Proposition 5.4.

(Item 2). Let M be a supported model of P and $\#$ a level numbering w.r.t. M. The uniqueness of $\#$ follows by Proposition 5.4. It follows that $M \models P$ and $M \models P^M$. Thus it is immediately clear that $\mathrm{LM}(P^M)$ is contained in M. It remains to prove that $M \subseteq \mathrm{LM}(P^M)$. We use complete induction on $\#\mathsf{a} > 1$ to show that $\mathsf{a} \in M$ implies $\mathsf{a} \in \mathrm{LM}(P^M)$.

Base case: $\#\mathsf{a} = 1$. Suppose that $\mathsf{a} \in M$. Since $\#\mathsf{a} = 1$, the only possibility is that there is a rule $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$ and $\mathrm{body}^+(r) = \emptyset$. It follows that $\mathsf{a} \leftarrow$ belongs to P^M so that $\mathsf{a} \in \mathrm{LM}(P^M)$.

Induction step: $\#\mathsf{a} = n > 1$. Suppose that $\mathsf{a} \in M$. Since $\#\mathsf{a} > 1$, there is a rule $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$, $\mathrm{body}^+(r) \neq \emptyset$, and $\#\mathsf{a} = \#r$. Then consider any $\mathsf{b} \in \mathrm{body}^+(r)$. Since $\#r = \max\{\#\mathsf{b}' \mid \mathsf{b}' \in \mathrm{body}^+(r)\} + 1$, we obtain $\#\mathsf{b} < n$. Moreover $r \in \mathrm{SR}(P, M)$ implies that $\mathsf{b} \in M$. Thus $\mathsf{b} \in \mathrm{LM}(P^M)$ by the inductive hypothesis and we have established that $\mathrm{body}^+(r) \subseteq \mathrm{LM}(P^M)$. On the other hand, $r \in \mathrm{SR}(P, M)$ implies that $r^+ = \mathsf{a} \leftarrow \mathrm{body}^+(r) \in P^M$. It follows that $\mathsf{a} \in \mathrm{LM}(P^M)$. □

6 NON-MODULAR TRANSLATION FUNCTIONS

In Section 4, we showed that faithful and modular translations cannot be established between certain classes of logic programs. However, this does not exclude the possibility that a polynomial and faithful, but *non-modular* translation function could be devised for the classes involved. Such alternatives are taken into consideration in this section. We proceed as follows. Section 6.1 covers the case of positive programs in which non-modular alternatives are very easy to obtain. A non-modular and faithful translation function from normal programs to atomic normal programs is developed in Section 6.2. This is a far more complicated objective, as normal programs may possess multiple stable models. Finally, in analogy to Section 4, we end this section by comparing logic programs with sets of clauses in Section 6.3.

6.1 Positive Programs Revisited

Theorems 4.7 and 4.10, which were established in Section 4.2, state that $\mathcal{B}^+ \not\leq_{\mathrm{FM}} \mathcal{U}^+$ and $\mathcal{U}^+ \not\leq_{\mathrm{FM}} \mathcal{A}^+$, respectively. Despite these relationships, it is straightforward to obtain a non-modular translation in case of positive programs. Basically, this boils down to the fact that the least model $\mathrm{LM}(P)$ can be be computed in polynomial time for any $P \in \mathcal{P}^+$. Then it is possible to reduce any $P \in \mathcal{P}^+$ to an atomic program in the following way.

Definition 6.1 *For any positive program $P \in \mathcal{P}^+$, define*

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

Moreover, let $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{LM}}(P)) = \mathrm{Hb}(P) - \mathrm{LM}(P)$ *so that* $\mathrm{Hb}(\mathrm{Tr}_{\mathrm{LM}}(P)) = \mathrm{Hb}(P)$. *The visible part* $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{LM}}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$.

Theorem 6.2 $\mathcal{P}^+ \leq_{\mathrm{PF}} \mathcal{A}^+$.

PROOF. It is clear that $\mathrm{Tr}_{\mathrm{LM}}$ is faithful, since $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{LM}}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$ by definition and both P and $\mathrm{Tr}_{\mathrm{LM}}(P)$ have the unique stable model $\mathrm{LM}(P) = \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(P))$ by Lemma 4.9. The translation function $\mathrm{Tr}_{\mathrm{LM}}$ is polynomial, as $\mathrm{LM}(P)$ can be computed in polynomial time. If the iterative construction from Section 2.2 is used as a basis, a quadratic algorithm is obtained, but there is also a linear time algorithm available [9]. □

As a consequence, the relations $\leq_{\mathrm{PFM}}$ and $\leq_{\mathrm{FM}}$ give rise to diverse classifications for the classes of positive logic programs. In fact, the hierarchy obtained with $\leq_{\mathrm{PFM}}$ is more refined than the one obtained with $\leq_{\mathrm{FM}}$. Thus $\leq_{\mathrm{PFM}}$ seems to provide a more accurate measure of expressiveness.

Corollary 6.3 $\mathcal{A}^+ =_{\mathrm{PF}} \mathcal{U}^+ =_{\mathrm{PF}} \mathcal{B}^+ =_{\mathrm{PF}} \mathcal{P}^+$.

Example 6.4 The translation function $\mathrm{Tr}_{\mathrm{LM}}$ introduced in Definition 6.1 is indeed non-modular. For instance, the programs $U = \{\mathsf{a} \leftarrow \mathsf{b}\}$ and $A = \{\mathsf{b} \leftarrow\}$ from the proof of Theorem 4.10 are translated as follows: $\mathrm{Tr}_{\mathrm{LM}}(U \cup A) = \{\mathsf{a} \leftarrow;\ \mathsf{b} \leftarrow\}$, but $\mathrm{Tr}_{\mathrm{LM}}(U) = \emptyset$ with $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{LM}}(U)) = \{\mathsf{a}, \mathsf{b}\}$ and $\mathrm{Tr}_{\mathrm{LM}}(A) = \{\mathsf{b} \leftarrow\}$. Thus $\mathrm{Tr}_{\mathrm{LM}}(U \cup A) \neq \mathrm{Tr}_{\mathrm{LM}}(U) \cup \mathrm{Tr}_{\mathrm{LM}}(A)$. □

Generally speaking, the translation $\mathrm{Tr}_{\mathrm{NM}}(P)$ obtained by a *non-modular* translation function $\mathrm{Tr}_{\mathrm{NM}}$ is often heavily dependent on particular instances of P so that already slight changes to P may alter $\mathrm{Tr}_{\mathrm{NM}}(P)$ completely. Consequently, a shortcoming of non-modular translations is that they do not necessarily support *updates*. This is also clear on the basis of the translation function $\mathrm{Tr}_{\mathrm{LM}}$ introduced in Theorem 6.2. E.g., in Example 6.4, the effect of removing A from $\mathrm{Tr}_{\mathrm{LM}}(U \cup A) = \{\mathsf{a} \leftarrow;\ \mathsf{b} \leftarrow\}$ is thorough, as $\mathrm{Tr}_{\mathrm{LM}}(U) = \emptyset$.

6.2 Translating Normal Programs into Atomic Ones

As shown in Section 6.1, it is easy to obtain a non-modular and faithful translation function for removing positive body literals in the case of positive programs. The setting becomes far more complicated when normal logic programs are taken into consideration, since a normal program may possess several stable models and it is not clear how to apply $\mathrm{Tr}_{\mathrm{LM}}$ from Definition 6.1. Nevertheless, we intend to develop a polynomial and faithful translation function $\mathrm{Tr}_{\mathrm{AT}}$ so that an arbitrary normal program P gets translated into an atomic program $\mathrm{Tr}_{\mathrm{AT}}(P)$. It is clear by the results presented in Section 4.3 that $\mathrm{Tr}_{\mathrm{AT}}$ must be non-modular if faithfulness is to be expected. Our idea is to apply the characterization of stable models developed in Section 5 so that each stable model M of a normal program P is eventually captured as a supported model M of P possessing a level numbering w.r.t. M. To recall the basic concepts from Section 5 we give an example of a level numbering.

Example 6.5 Let $P = \{r_1, r_2, r_3\}$ be a (positive) normal program consisting of the rules $r_1 = \mathsf{a} \leftarrow;\ r_2 = \mathsf{a} \leftarrow \mathsf{b};$ and $r_3 = \mathsf{b} \leftarrow \mathsf{a}$ so that $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}(P) = \{\mathsf{a}, \mathsf{b}\}$. The unique stable model $M = \mathrm{LM}(P) = \{\mathsf{a}, \mathsf{b}\}$ is supported by $\mathrm{SR}(P, M) = P$. The unique level numbering $\#$ w.r.t. M is determined by $\#r_1 = 1$, $\#\mathsf{a} = 1$, $\#r_3 = 2$, $\#\mathsf{b} = 2$, and $\#r_2 = 3$. □

However, there is no explicit way of representing a level numbering within a normal program and we have to encode such a numbering using propositional atoms. Then a natural solution is to use a binary representation for the individual numbers determined by a level numbering $\#$. Unfortunately, every atom in $\mathrm{Hb}(P)$ may be assigned a different level number in the worst case. This setting is actually demonstrated in Example 6.5. Thus the level numbers of atoms may vary from 1 to $|\mathrm{Hb}(P)|$. Hence the highest possible level number of a rule $r \in P$ is $|\mathrm{Hb}(P)| + 1$, as for r_2 in our example. Although level numbers are positive numbers by definition, we leave room for 0 which is to act as the least binary value. Thus, given a normal program P, we have to be prepared for binary numbers consisting of at most

$$\nabla P = \lceil \log_2(|\mathrm{Hb}(P)| + 2) \rceil \tag{6.1}$$

bits. In case of Example 6.5, we have $\nabla P = 2$ which is enough to represent all the values in the range of the level numbering $\#$ in question. In general, we can establish the following bounds for level numbers in terms of ∇P.

Proposition 6.6 *If M is a supported model of a normal program P and $\# : M \cup \mathrm{SR}(P, M) \to \mathbb{N}$ a level numbering w.r.t. M, then $0 < \#\mathsf{a} < 2^{\nabla P} - 1$ for every $\mathsf{a} \in M$ and $0 < \#r < 2^{\nabla P}$ for every $r \in \mathrm{SR}(P, M)$.*

PROOF. Let us consider the lower bounds first. The level number $\#r \in \mathbb{N}$ assigned to a rule $r \in \mathrm{SR}(P, M)$ is necessarily greater than zero, because $\#r$ is defined either (i) as a maximum among a set of natural numbers *increased by one* or (ii) as 1. Thus the level number $\#\mathsf{a}$ assigned to any atom $\mathsf{a} \in M$ greater than 0, as $\#\mathsf{a}$ is defined as a minimum of a set of natural numbers that are greater than 0. To show the upper bounds for level numbers, we use complete induction on natural numbers $n \geq 1$ in order to establish that

1. if $\#\mathsf{a} = n$ for some $\mathsf{a} \in M$, then $\#\mathsf{a} < 2^{\nabla P} - 1$ and there is a sequence of atoms $\mathsf{a}_1, \ldots \mathsf{a}_n$ from M such that $\#\mathsf{a}_i = i$ for all $i \in \{1, \ldots, n\}$ and $\mathsf{a}_n = \mathsf{a}$; and

2. if $\#r = n$ for some $r \in \mathrm{SR}(P, M)$, then $\#r < 2^{\nabla P}$.

In the base case, $\#\mathsf{a} = 1$ for an atom $\mathsf{a} \in M$ or $\#r = 1$ for a rule $r \in \mathrm{SR}(P, M)$. In both cases, $\mathrm{Hb}(P)$ contains at least one atom, so that $\nabla P \geq 2$. Thus $\#\mathsf{a} < 3 \leq 2^{\nabla P} - 1$ and $\#r < 4 \leq 2^{\nabla P}$. Moreover, if $\#\mathsf{a} = 1$ for some atom $\mathsf{a} \in M$, then the sequence involved in the first item contains a alone.

Induction step follows. The second claim is proved first. Suppose that $\#r = n > 1$ holds for some rule $r \in \mathrm{SR}(P, M)$. Then $\mathrm{body}^+(r) \neq \emptyset$ holds necessarily by Definition 5.2 and $\#r$ equals to $\max\{\#\mathsf{b} \mid \mathsf{b} \in \mathrm{body}^+(r)\} + 1$. Thus $\#\mathsf{b} < n$ holds for each $\mathsf{b} \in \mathrm{body}^+(r)$ so that we can conclude $\max\{\#\mathsf{b} \mid \mathsf{b} \in \mathrm{body}^+(r)\} < 2^{\nabla P} - 1$ by the inductive hypothesis. Therefore $\#r < 2^{\nabla P}$.

Then suppose that $\#\mathsf{a} = n$ for some $\mathsf{a} \in M$. By Definition 5.2, there is a rule $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$ and $\#r = n > 1$, too. The definition of $\#r$ implies the existence of an atom $\mathsf{b} \in \mathrm{body}^+(r)$ such that $\#\mathsf{b} = n - 1$. It follows by the inductive hypothesis that there is a sequence of atoms $\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}$ from M such that $\mathsf{a}_{n-1} = \mathsf{b}$ and $\#\mathsf{a}_i = i$ for all $i \in \{1, \ldots, n-1\}$. By adding a in the end of this sequence, we obtain a sequence $\mathsf{a}_1, \ldots, \mathsf{a}_n$ of atoms from M such that $\#\mathsf{a}_i = i$ for all $i \in \{1, \ldots, n\}$ and $\mathsf{a}_n = \mathsf{a}$. If n is to be maximal, then $\mathsf{a}_1, \ldots, \mathsf{a}_n$ must enumerate $\mathrm{Hb}(P)$ as whole, i.e. $\#\mathsf{a} = n = |\mathrm{Hb}(P)|$. Then recall the definition of ∇P. As $\mathsf{a} \in \mathrm{Hb}(P)$, we have $|\mathrm{Hb}(P)| \geq 1$ so that $1 \leq \log_2(|\mathrm{Hb}(P)|+1) < \log_2(|\mathrm{Hb}(P) + 2|) \leq \lceil \log_2(|\mathrm{Hb}(P)| + 2) \rceil$. The respective powers of 2 yield $|\mathrm{Hb}(P)| + 1 < |\mathrm{Hb}(P) + 2| \leq 2^{\nabla P}$ which implies $|\mathrm{Hb}(P)| < 2^{\nabla P} - 1$. It follows that $\#\mathsf{a} = |\mathrm{Hb}(P)| < 2^{\nabla P} - 1$ as desired. □

The logarithmic factor embodied in ∇P forms an important design criterion for us, since would like to keep the length of the translation $||\mathrm{Tr}_{\mathrm{AT}}(P)||$ as well as the translation time proportional to $||P|| \times \nabla P$ rather than $||P|| \times |\mathrm{Hb}(P)|$. Hence we strive for a sub-quadratic translation function from $\mathcal{P}$ to $\mathcal{A}$. To get an idea of the potential behind such an objective, we have $\nabla P = 14$ for programs P with $|\mathrm{Hb}(P)| = 10000$, for instance.

Representing Binary Counters

We have to fix some notation in order to deal with binary representations of natural numbers. Given the number of bits b and a natural number $0 \leq n < 2^b$, we write $n[i \ldots j]$, where $0 < i \leq j \leq b$, for the binary representation of n from the i^{th} bit to the j^{th} bit in the decreasing order of significance. Thus

$n[1 \ldots b]$ gives a complete binary representation for n. Moreover, as a special case of this notation, we may refer to the i^{th} bit by writing $n[i] = n[i \ldots i]$.

Technically speaking, the idea is to encode the level number $\#\mathsf{a}$ for a particular atom $\mathsf{a} \in \mathrm{Hb}(P)$ using a *vector* $\mathsf{a}_1, \ldots, \mathsf{a}_j$ of new atoms where $j = \nabla P$. Such a vector can be understood as a representation of a *binary counter* of j bits; the first and the last atoms corresponding to the most significant and the least significant bits, respectively. The idea is to equate bits 0 and 1 with the truth values false and true assigned to atoms, since atoms may take only two values under the stable model semantics. Because the forthcoming translation is supposed to be an atomic normal program, positive body literals are forbidden and we have to introduce the vector $\overline{\mathsf{a}_1}, \ldots, \overline{\mathsf{a}_j}$ of complementary atoms so that we can condition rules on both values of bits. This is exactly the technique that was demonstrated in Example 4.20. In the current setting, the idea is that the i^{th} bit of the binary counter associated with the atom a takes the value 0 (resp. 1) if and only if a_i (resp. $\overline{\mathsf{a}_i}$) cannot be inferred, i.e. the negative literal ${\sim}\mathsf{a}_i$ (resp. ${\sim}\overline{\mathsf{a}_i}$) is satisfied in rule bodies. In the sequel, we may introduce a binary counter of the kind above for any atom a by subscripting it with an index i in the range $0 < i \leq j$. Such a representation involves a set of new atoms $\mathrm{Hb}^{\mathrm{ctr}}_j(\mathsf{a})$ defined below.

Definition 6.7 *Given a number of bits j and an atom a, let*

$$\mathrm{Hb}^{\mathrm{ctr}}_j(\mathsf{a}) = \{\mathsf{a}_i, \overline{\mathsf{a}_i} \mid 0 < i \leq j\}. \tag{6.2}$$

In order to express the constraints on level numberings, as demanded by Definition 5.2, we need certain primitive operations on binary counters. These primitives will be used as subprograms of the forthcoming translation $\mathrm{Tr}_{\mathrm{AT}}(P)$ for normal programs P. The first set of subprograms, as listed in Table 2, concentrates on setting the counters to particular values. The size of each subprogram is governed by a parameter j which gives the number of bits used in the binary counters involved. The activation of all subprograms is controlled by an additional atom c. The idea is that the respective subprograms are activated only when c cannot be inferred, i.e. c is assigned to false under stable model semantics. In the following, we give brief descriptions of the first three primitives.

1. The subprogram $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})$ selects a value between 0 and $2^j - 1$ for the binary counter $\mathsf{a}_1, \ldots, \mathsf{a}_j$ associated with an atom a.

2. The program $\mathrm{NXT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})$ binds the values of the binary counters associated with atoms a and b, respectively, so that the latter is the former increased by one (modulo 2^j). We have chosen ∇P big enough so that counters to be used in the sequel do not wrap in practice.

3. The last subprogram $\mathrm{FIX}_j(\mathsf{a}, n, \mathsf{c})$ assigns a fixed value $0 \leq n < 2^j$, in the binary representation, to the counter associated with the atom a.

In addition to setting the values of counters, we have to be able to compare them. Table 3 lists our basic primitives in this respect: implementations of the relations $<$ and $=$ as atomic normal programs. Our explanations follow.

Primitive	Definition
$\mathrm{SEL}_j(\mathsf{a},\mathsf{c})$	$= \{\mathsf{a}_i \leftarrow \sim\overline{\mathsf{a}_i}, \sim\mathsf{c};\ \ \overline{\mathsf{a}_i} \leftarrow \sim\mathsf{a}_i, \sim\mathsf{c} \mid 0 < i \leq j\}$
$\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$	$= \{\overline{\mathsf{b}_i} \leftarrow \sim\overline{\mathsf{a}_i}, \sim\overline{\mathsf{a}_{i+1}}, \sim\mathsf{b}_{i+1}, \sim\mathsf{c} \mid 0 < i < j\} \cup$
	$\{\mathsf{b}_i \leftarrow \sim\mathsf{a}_i, \sim\overline{\mathsf{a}_{i+1}}, \sim\mathsf{b}_{i+1}, \sim\mathsf{c} \mid 0 < i < j\} \cup$
	$\{\overline{\mathsf{b}_i} \leftarrow \sim\mathsf{a}_i, \sim\mathsf{a}_{i+1}, \sim\mathsf{c} \mid 0 < i < j\} \cup$
	$\{\mathsf{b}_i \leftarrow \sim\overline{\mathsf{a}_i}, \sim\mathsf{a}_{i+1}, \sim\mathsf{c} \mid 0 < i < j\} \cup$
	$\{\overline{\mathsf{b}_i} \leftarrow \sim\mathsf{a}_i, \sim\overline{\mathsf{b}_{i+1}}, \sim\mathsf{c} \mid 0 < i < j\} \cup$
	$\{\mathsf{b}_i \leftarrow \sim\overline{\mathsf{a}_i}, \sim\overline{\mathsf{b}_{i+1}}, \sim\mathsf{c} \mid 0 < i < j\} \cup$
	$\{\overline{\mathsf{b}_j} \leftarrow \sim\overline{\mathsf{a}_j}, \sim\mathsf{c};\ \ \mathsf{b}_j \leftarrow \sim\mathsf{a}_j, \sim\mathsf{c}\}$
$\mathrm{FIX}_j(\mathsf{a},n,\mathsf{c})$	$= \{\overline{\mathsf{a}_i} \leftarrow \sim\mathsf{c} \mid 0 < i \leq j \text{ and } n[i] = 0\} \cup$
	$\{\mathsf{a}_i \leftarrow \sim\mathsf{c} \mid 0 < i \leq j \text{ and } n[i] = 1\}$

Table 2: Primitives for selecting the values of binary counters

4. The program $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$ checks if the value of the binary counter associated with an atom a is strictly lower than the value of the binary counter associated with another atom b. To keep the program linear in j, we need a vector of new atoms $\mathsf{lt}(\mathsf{a},\mathsf{b})_1, \ldots, \mathsf{lt}(\mathsf{a},\mathsf{b})_j$ plus the corresponding vector of complementary atoms which we associate with a and b. The atoms $\mathsf{lt}(\mathsf{a},\mathsf{b})_1$ and $\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_1}$, which refer to the most significant bits, capture the result of the comparison. In particular, note that $\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_1}$ means intuitively that the counter associated with the atom a holds a value that is greater than or equal to the one held by the counter associated with the atom b. This relation will also be needed when comparing the values of counters.

5. Quite similarly, the program $\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})$ checks if the counters associated with the atoms a and b hold the same value. In this case, only two new atoms $\mathsf{eq}(\mathsf{a},\mathsf{b})$ and $\overline{\mathsf{eq}(\mathsf{a},\mathsf{b})}$, which capture the result of the comparison, are needed.

Our next goal is to specify the expected outcomes of the primitives listed in Tables 2 and 3. However, the correctness proofs are postponed until Section 9. Whenever the value of a counter of j bits associated with an atom a is chosen to be $0 \leq n < 2^j$, the contribution of the respective program $\mathrm{SEL}_j(\mathsf{a},\mathsf{c})$ is a set of atoms

$$\mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n) = \{\mathsf{a}_i \mid 0 < i \leq j \text{ and } n[i] = 1\} \cup \{\overline{\mathsf{a}_i} \mid 0 < i \leq j \text{ and } n[i] = 0\} \tag{6.3}$$

given that the atom c is not inferable. The reader might ponder the reference to the value n at this point, as the *choice* of n is to be made. But when $\mathrm{SEL}_j(\mathsf{a},\mathsf{c})$ is used as a subprogram, the stability condition from Definition

Primitive	Definition
$\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c}) =$	$\{\mathsf{lt}(\mathsf{a},\mathsf{b})_i \leftarrow \sim\mathsf{a}_i, \sim\overline{\mathsf{b}_i}, \sim\mathsf{c} \mid 0 < i \leq j\} \cup$
	$\{\mathsf{lt}(\mathsf{a},\mathsf{b})_i \leftarrow \sim\mathsf{a}_i, \sim\mathsf{b}_i, \sim\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_{i+1}}, \sim\mathsf{c} \mid 0 < i < j\} \cup$
	$\{\mathsf{lt}(\mathsf{a},\mathsf{b})_i \leftarrow \sim\overline{\mathsf{a}_i}, \sim\overline{\mathsf{b}_i}, \sim\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_{i+1}}, \sim\mathsf{c} \mid 0 < i < j\} \cup$
	$\{\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_i} \leftarrow \sim\mathsf{lt}(\mathsf{a},\mathsf{b})_i, \sim\mathsf{c} \mid 0 < i \leq j\}$
$\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c}) =$	$\{\overline{\mathsf{eq}(\mathsf{a},\mathsf{b})} \leftarrow \sim\mathsf{a}_i, \sim\overline{\mathsf{b}_i}, \sim\mathsf{c} \mid 0 < i \leq j\} \cup$
	$\{\overline{\mathsf{eq}(\mathsf{a},\mathsf{b})} \leftarrow \sim\overline{\mathsf{a}_i}, \sim\mathsf{b}_i, \sim\mathsf{c} \mid 0 < i \leq j\} \cup$
	$\{\mathsf{eq}(\mathsf{a},\mathsf{b}) \leftarrow \sim\overline{\mathsf{eq}(\mathsf{a},\mathsf{b})}, \sim\mathsf{c}\}$

Table 3: Primitives for comparing the values of binary counters

2.2 implies a fixed point condition on n so that any value of n in the range from 0 to $2^j - 1$ is possible. The effect a subprogram $\mathrm{NXT}_j(\mathsf{b},\mathsf{a},\mathsf{c})$ is supposed to be the same set of atoms (6.3) given that the counter associated with some other atom b is holding a value m such that $n = m + 1 \bmod 2^j$.

In analogy to the preceding two subprograms, we have to define the result of a subprogram $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$ when the atom c is assigned to false. It is assumed in (6.4) below that the values of the counters associated with the atoms a and b are n and m in the ranges $0 \leq n < 2^j$ and $0 \leq m < 2^j$, respectively. Then let

$$\begin{aligned}(6.4)\quad \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m) = \\ \{\mathsf{lt}(\mathsf{a},\mathsf{b})_i \mid 0 < i \leq j \text{ and } n[i\ldots j] < m[i\ldots j]\} \cup \\ \{\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_i} \mid 0 < i \leq j \text{ and } n[i\ldots j] \geq m[i\ldots j]\}.\end{aligned}$$

Note that c is neglected in (6.4), as no atom can be inferred by the rules of $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$ when c is inferable; i.e. assigned to true in a stable model. The result of testing the equality of the counters is defined analogously. The outcome for $\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})$, when c is not inferable, is the following:

$$(6.5)\qquad \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},m) = \{\mathsf{eq}(\mathsf{a},\mathsf{b}) \mid n = m\} \cup \{\overline{\mathsf{eq}(\mathsf{a},\mathsf{b})} \mid n \neq m\}.$$

On the other hand, given an atom a and a set of atoms N — such as a stable model of a program involving the subprograms under consideration — we may *extract* the value of the counter $\mathsf{a}_1,\ldots,\mathsf{a}_j$ associated with an atom a by

$$(6.6)\qquad \mathrm{val}_j(\mathsf{a},N) = \sum\{2^{j-i} \mid 0 < i \leq j \text{ and } \mathsf{a}_i \in N\}.$$

It follows that $\mathrm{val}_j(\mathsf{a},N)[i] = 1 \iff \mathsf{a}_i \in N$ holds for each $0 < i \leq j$. Moreover, we have $\mathrm{val}_j(\mathsf{a},\mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n)) = n$ for any n for which $0 \leq n < 2^j$.

A Non-Modular Translation Function $\mathrm{Tr}_{\mathrm{AT}}$

In the sequel, we will compose a non-modular translation function $\mathrm{Tr}_{\mathrm{AT}}$ in four steps corresponding Definitions 6.8–6.11 to be presented. In each definition, we specify a subtranslation (say $\mathrm{Tr}_{\mathrm{SUB}}(P)$) of $\mathrm{Tr}_{\mathrm{AT}}(P)$ as well as the

set $\mathrm{head}(\mathrm{Tr}_{\mathrm{SUB}}(P))$, which determines the atoms effectively *defined* by the subtranslation $\mathrm{Tr}_{\mathrm{SUB}}(P)$ in question. These sets will be disjoint for the four subtranslations to be presented. To achieve faithfulness, one of the aims is to capture each stable model M of a normal logic program P as a stable model N of $\mathrm{Tr}_{\mathrm{AT}}(P)$ which is an atomic program. In the subsequent discussion, M and N are supposed to form a pair of stable models in a one-to-one correspondence, as insisted by faithfulness. The first part of the translation $\mathrm{Tr}_{\mathrm{SUPP}}(P)$ aims to capture a supported model M of the program P and to define the complementary atom $\overline{\mathsf{a}}$ for each atom $\mathsf{a} \in \mathrm{Hb}(P)$.

Definition 6.8 *For a normal program P, define an atomic normal program*

$$\begin{aligned}(6.7)\qquad \mathrm{Tr}_{\mathrm{SUPP}}(P) = {} & \{\overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a} \mid \mathsf{a} \in \mathrm{Hb}(P)\} \cup \\ & \{\mathsf{bt}(r) \leftarrow {\sim}\overline{\mathrm{body}^+(r)}, {\sim}\mathrm{body}^-(r) \mid r \in P\} \cup \\ & \{\overline{\mathsf{bt}(r)} \leftarrow {\sim}\mathsf{bt}(r) \mid r \in P\} \cup \\ & \{\mathrm{head}(r) \leftarrow {\sim}\overline{\mathsf{bt}(r)} \mid r \in P\}.\end{aligned}$$

The set of atoms $\mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ *is*

$$(6.8)\qquad \mathrm{head}(P) \cup \{\overline{\mathsf{a}} \mid \mathsf{a} \in \mathrm{Hb}(P)\} \cup \{\mathsf{bt}(r), \overline{\mathsf{bt}(r)} \mid r \in P\}.$$

In principle, it would be sufficient to rewrite a rule $r \in P$ as

$$(6.9)\qquad \mathrm{head}(r) \leftarrow {\sim}\overline{\mathrm{body}^+(r)}, {\sim}\mathrm{body}^-(r),$$

but other parts of the overall translation require us to determine when the *body* of r is *true*. This is why new atoms $\mathsf{bt}(r)$ and $\overline{\mathsf{bt}(r)}$ are introduced for each $r \in P$. Note that copying the transformed body of r to other parts of the translation would imply a quadratic blow-up and we need $\mathsf{bt}(r)$ for each $r \in P$ in order to save space. The next part of the translation introduces counters that are needed to represent a level numbering candidate.

Definition 6.9 *For a normal program P, define an atomic normal program*

$$\begin{aligned}(6.10)\quad \mathrm{Tr}_{\mathrm{CTR}}(P) = {} & \bigcup_{\mathsf{a} \in \mathrm{Hb}(P)} [\mathrm{SEL}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), \overline{\mathsf{a}}) \cup \mathrm{NXT}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), \mathsf{nxt}(\mathsf{a}), \overline{\mathsf{a}})] \cup \\ & \bigcup_{r \in P \text{ and } \mathrm{body}^+(r) = \emptyset} \mathrm{FIX}_{\nabla P}(\mathsf{ctr}(r), 1, \overline{\mathsf{bt}(r)}) \cup \\ & \bigcup_{r \in P \text{ and } \mathrm{body}^+(r) \neq \emptyset} \mathrm{SEL}_{\nabla P}(\mathsf{ctr}(r), \overline{\mathsf{bt}(r)}).\end{aligned}$$

Consequently, the set of atoms

$$\begin{aligned}(6.11)\quad \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P)) = {} & \bigcup_{\mathsf{a} \in \mathrm{Hb}(P)} [\mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(\mathsf{a})) \cup \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt}(\mathsf{a}))] \cup \\ & \bigcup_{r \in P \text{ and } \mathrm{body}^+(r) = \emptyset} \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), 1) \cup \\ & \bigcup_{r \in P \text{ and } \mathrm{body}^+(r) \neq \emptyset} \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r)).\end{aligned}$$

In this way, two new atoms $\mathsf{ctr}(\mathsf{a})$ and $\mathsf{nxt}(\mathsf{a})$, which act as names of two counters, are introduced for each atom $\mathsf{a} \in \mathrm{Hb}(P)$. The eventual purpose of these counters is to hold the values $\#\mathsf{a}$ and $\#\mathsf{a} + 1$, respectively, in case that a belongs to the domain of a level numbering $\#$, i.e. $\mathsf{a} \in M$; or equivalently, $\overline{\mathsf{a}} \notin N$. However, at this point, the primitives included in $\mathrm{Tr}_{\mathrm{CTR}}(P)$ choose a value for $\mathsf{ctr}(\mathsf{a})$ and define the value of $\mathsf{nxt}(\mathsf{a})$ as the successor of the value of $\mathsf{ctr}(\mathsf{a})$ modulo $2^{\nabla P}$. Quite similarly, a new atom $\mathsf{ctr}(r)$ and the respective counter is introduced for each $r \in P$ to eventually hold $\#r$ when r is in the domain of $\#$, i.e. $r \in \mathrm{SR}(P, M)$, or equivalently $\overline{\mathsf{bt}(r)} \notin N$. In case of an atomic rule $r \in P$ with $\mathrm{body}^+(r) = \emptyset$, the counter $\mathsf{ctr}(r)$ is assigned a fixed value 1 and no choice is made. Note that such a special restriction is in accordance with Definition 5.2.

The translation $\mathrm{Tr}_{\mathrm{CTR}}(P)$ is sufficient for choosing a candidate level numbering for a supported model M of P that is to be captured by the rules in $\mathrm{Tr}_{\mathrm{SUPP}}(P)$. We have to introduce constraints in order to ensure that the candidate is indeed a level numbering, as dictated by Definition 5.2. We start with the conditions imposed on rules $r \in P$ and in particular, when $r \in \mathrm{SR}(P, M)$ holds, i.e. $M \models \mathrm{body}(r)$. This explains why $\overline{\mathsf{bt}(r)}$ is used as a controlling atom in the forthcoming translation. As explained above, the case of atomic rules $r \in P$ with $\mathrm{body}^+(r) = \emptyset$ is already covered by $\mathrm{Tr}_{\mathrm{CTR}}(P)$ which assigns a fixed value — the natural number 1 — to $\mathsf{ctr}(r)$. But for non-atomic rules $r \in P$ with $\mathrm{body}^+(r) \neq \emptyset$, the maximization principle from Definition 5.2 must be expressed e.g. as follows.

Definition 6.10 *Let* x *be a new atom not appearing in* $\mathrm{Hb}(P)$. *For an non-atomic rule* $r \in P$ *and a number of bits* b, *define*

$$\mathrm{Tr}_{\mathrm{MAX}}(r, b) = \bigcup_{\mathsf{a} \in \mathrm{body}^+(r)} \mathrm{Tr}_{\mathrm{MAX}}(r, b, \mathsf{a}) \tag{6.12}$$

where for any $\mathsf{a} \in \mathrm{body}^+(r)$, $\mathrm{Tr}_{\mathrm{MAX}}(r, b, \mathsf{a}) =$

$$\begin{aligned}
&\mathrm{LT}_b(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}), \overline{\mathsf{bt}(r)}) \cup \\
&\mathrm{EQ}_b(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}), \overline{\mathsf{bt}(r)}) \cup \\
&\{\mathsf{x} \leftarrow {\sim}\mathsf{x}, {\sim}\overline{\mathsf{bt}(r)}, {\sim}\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))_1}\} \cup \\
&\{\mathsf{max}(r) \leftarrow {\sim}\overline{\mathsf{bt}(r)}, {\sim}\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))}\}.
\end{aligned}$$

For a normal program P, *define an atomic normal program*

$$\begin{aligned}
\mathrm{Tr}_{\mathrm{MAX}}(P) = &\bigcup_{r \in P \text{ and } \mathrm{body}^+(r) \neq \emptyset} \mathrm{Tr}_{\mathrm{MAX}}(r, \nabla P) \cup \\
&\{\mathsf{x} \leftarrow {\sim}\mathsf{x}, {\sim}\overline{\mathsf{bt}(r)}, {\sim}\mathsf{max}(r) \mid r \in P \text{ and } \mathrm{body}^+(r) \neq \emptyset\}.
\end{aligned} \tag{6.13}$$

Consequently, the set of atoms

$$\begin{aligned}
\mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P)) = &\{\mathsf{x}\} \cup \{\mathsf{max}(r) \mid r \in P \text{ and } \mathrm{body}^+(r) \neq \emptyset\} \cup \\
&\bigcup_{r \in P \text{ and } \mathsf{a} \in \mathrm{body}^+(r)} \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))) \cup \\
&\{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a})), \overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))} \mid r \in P \text{ and } \mathsf{a} \in \mathrm{body}^+(r)\}.
\end{aligned} \tag{6.14}$$

An informal description follows. The rules in $\mathrm{Tr}_{\mathrm{MAX}}(r, \nabla P, \mathsf{a})$ are to be activated for a non-atomic rule $r \in \mathrm{SR}(P, M)$ and a positive body atom $\mathsf{a} \in \mathrm{body}^+(r)$. As a consequence, the value held by $\mathsf{ctr}(r)$ must be greater than or equal to the value of $\mathsf{nxt}(\mathsf{a})$ which is supposed to be the value of $\mathsf{ctr}(\mathsf{a})$ increased by one. In addition to this, the rules for $\mathsf{max}(r)$ in $\mathrm{Tr}_{\mathrm{MAX}}(r, \nabla P, \mathsf{a})$ and $\mathrm{Tr}_{\mathrm{MAX}}(P)$ make the value of $\mathsf{ctr}(r)$ equal to the value of $\mathsf{nxt}(\mathsf{a})$ for some $\mathsf{a} \in \mathrm{body}^+(r)$. Thus the value of $\mathsf{ctr}(r)$ must be the maximum among the values of the counters $\mathsf{nxt}(\mathsf{a})$ associated with the positive body atoms $\mathsf{a} \in \mathrm{body}^+(r)$. This conforms to the definition of $\#r$ given in Definition 5.2.

Let us then turn our attention to atoms a that are assigned to true in a supported model M of P. By Definition 5.1 such an atom must have a rule $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$. Moreover, the level number $\#\mathsf{a}$ is defined as the minimum among the respective rules by Definition 5.2.

Definition 6.11 *Let* y *be a new atom not appearing in* $\mathrm{Hb}(P)$*. For a rule* r *and a number of bits* b*, define* $\mathrm{Tr}_{\mathrm{MIN}}(r, b) =$

$$\begin{aligned}
&\mathrm{LT}_b(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r)), \overline{\mathsf{bt}(r)}) \cup \\
&\mathrm{EQ}_b(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r)), \overline{\mathsf{bt}(r)}) \cup \\
&\{\mathsf{y} \leftarrow \sim\mathsf{y}, \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r)))_1}\} \cup \\
&\{\mathsf{min}(\mathrm{head}(r)) \leftarrow \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r)))}\}.
\end{aligned}$$

For a normal program P*, define an atomic normal program*

$$\begin{aligned}
(6.15)\quad \mathrm{Tr}_{\mathrm{MIN}}(P) = \bigcup_{r \in P} \mathrm{Tr}_{\mathrm{MIN}}(r, \nabla P) \cup \\
\{\mathsf{y} \leftarrow \sim\mathsf{y}, \sim\overline{\mathsf{a}}, \sim\mathsf{min}(\mathsf{a}) \mid \mathsf{a} \in \mathrm{Hb}(P)\}.
\end{aligned}$$

Then the set of atoms

$$\begin{aligned}
(6.16)\quad \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P)) = \{\mathsf{y}\} \cup \{\mathsf{min}(\mathsf{a}) \mid \mathsf{a} \in \mathrm{head}(P)\} \cup \\
\bigcup_{r \in P} \mathrm{Hb}^{\mathsf{ctr}}_{\nabla P}(\mathsf{lt}(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r)))) \cup \\
\{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r))), \overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r)))} \mid r \in P\}.
\end{aligned}$$

Given $\mathsf{a} \in M$ and a rule $r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$, the rules in $\mathrm{Tr}_{\mathrm{MIN}}(r, \nabla P)$ make the value of $\mathsf{ctr}(\mathsf{a})$ lower than or equal to the value of $\mathsf{ctr}(r)$. Moreover, the rules for $\mathsf{min}(\mathsf{a})$ in $\mathrm{Tr}_{\mathrm{MIN}}(P)$ ensure that the value of $\mathsf{ctr}(\mathsf{a})$ equals to the value of $\mathsf{ctr}(r)$ for at least one such rule r. In this way, the value of $\mathsf{ctr}(\mathsf{a})$ becomes necessarily the minimum, which is in harmony with the definition of $\#\mathsf{a}$ in Definition 5.2. We are now ready to formulate $\mathrm{Tr}_{\mathrm{AT}}$ which is based on the four sub-translations presented in Definitions 6.8–6.11.

Definition 6.12 *Given a normal program* P*, define an atomic normal program*

$$\mathrm{Tr}_{\mathrm{AT}}(P) = \mathrm{Tr}_{\mathrm{SUPP}}(P) \cup \mathrm{Tr}_{\mathrm{CTR}}(P) \cup \mathrm{Tr}_{\mathrm{MAX}}(P) \cup \mathrm{Tr}_{\mathrm{MIN}}(P)$$

and $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{AT}}(P)) = \emptyset$*, as* $\mathrm{Hb}(P) \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{AT}}(P)) =$

$$\mathrm{Hb}(\mathrm{Tr}_{\mathrm{SUPP}}(P)) \cup \mathrm{Hb}(\mathrm{Tr}_{\mathrm{CTR}}(P)) \cup \mathrm{Hb}(\mathrm{Tr}_{\mathrm{MIN}}(P)) \cup \mathrm{Hb}(\mathrm{Tr}_{\mathrm{MAX}}(P)).$$

The visible part $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(P))$ *is defined as* $\mathrm{Hb}_{\mathrm{v}}(P)$.

By inspecting the four parts of $\mathrm{Tr}_{\mathrm{AT}}(P)$ once more, we note that $\mathrm{Tr}_{\mathrm{AT}}(P)$ can be formed in a very systematic fashion by generating certain rules for each $r \in P$ and $\mathsf{a} \in \mathrm{Hb}(P)$. However, $\mathrm{Tr}_{\mathrm{AT}}$ is not modular in the sense defined in Section 3.3. A source of non-modularity is hidden in the number of bits ∇P involved in $\mathrm{Tr}_{\mathrm{AT}}(P)$. Given two programs P and Q satisfying module conditions M1–M4, it is still possible that $\nabla P < \nabla(P \cup Q)$ and $\nabla Q < \nabla(P \cup Q)$. As a consequence, the counters involved in $\mathrm{Tr}_{\mathrm{AT}}(P)$ and $\mathrm{Tr}_{\mathrm{AT}}(Q)$ are based on too few bits, which implies that $\mathrm{Tr}_{\mathrm{AT}}(P)$ and $\mathrm{Tr}_{\mathrm{AT}}(Q)$ cannot be joined together in order to form the translation $\mathrm{Tr}_{\mathrm{AT}}(P \cup Q)$.

Example 6.13 Due to the number of rules generated by $\mathrm{Tr}_{\mathrm{AT}}$, let us consider a logic program P which contains only one rule $r = \mathsf{a} \leftarrow \mathsf{a}$. To give a better idea of the rules included in $\mathrm{Tr}_{\mathrm{AT}}(P)$, a complete listing of the translation is given in Figure 3 on page 54. Note that atoms are written in a flat notation to enable computations with SMODELS [41]. Moreover, complementary atoms (such as $\overline{\mathsf{a}}$ for a) are prefixed with "`not_`". Using these principles, e.g. the atom $\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))_1}$ is rewritten as `not_lt_ctr_r_nxt_a_1`. We may now use SMODELS to compute the only stable model of the translation:

```
smodels version 2.27. Reading...done
Answer: 1
Stable Model: not_bt not_a
False
```

In our previous notation, this is a stable model $N = \{\overline{\mathsf{bt}(r)}, \overline{\mathsf{a}}\}$ of $\mathrm{Tr}_{\mathrm{AT}}(P)$ which corresponds to the only stable model $M = \emptyset$ of P. Note that the rest of the translation ($\mathrm{Tr}_{\mathrm{CTR}}(P)$, $\mathrm{Tr}_{\mathrm{MAX}}(P)$, and $\mathrm{Tr}_{\mathrm{MIN}}(P)$) is inactive given that $\overline{\mathsf{bt}(r)}$ and $\overline{\mathsf{a}}$ are true. The last line of the output indicates that $\mathrm{Tr}_{\mathrm{AT}}(P)$ does not have further stable models; nor does P. However, if we drop the last three lines from the translation given in Figure 3, we obtain four additional stable models. One of them assigns true atoms that are listed below:

```
a bt_r
not_ctr_a_1 ctr_a_2
nxt_a_1 not_nxt_a_2
ctr_r_1 not_ctr_r_2
not_lt_ctr_r_nxt_a_1 not_lt_ctr_r_nxt_a_2 eq_ctr_r_nxt_a
max_r
not_lt_ctr_r_ctr_a_1 lt_ctr_r_ctr_a_2 not_eq_ctr_r_ctr_a
```

According to this stable model, say N', we have a situation in which a evaluates to true. This implies that the body of r is satisfied, i.e. $\mathsf{bt}(r)$ is true. The counters $\mathsf{ctr}(\mathsf{a})$, $\mathsf{nxt}(\mathsf{a})$, and $\mathsf{ctr}(r)$ hold the values $01_2 = 1$, $10_2 = 2$, and $10_2 = 2$, respectively. Moreover, the value held by $\mathsf{ctr}(r)$ is detected to be (i) greater than or equal to the value held by $\mathsf{nxt}(\mathsf{a})$ and (ii) equal to the value held by $\mathsf{nxt}(\mathsf{a})$. Thus $\mathsf{ctr}(r)$ holds a maximum value, which is indicated by

$\mathsf{max}(r)$ being true. Finally, the value held by $\mathsf{ctr}(r)$ is not lower than that held by $\mathsf{ctr}(\mathsf{a})$. However, the values are not equal, which indicates that the value held by $\mathsf{ctr}(\mathsf{a})$ is not appropriate as a minimum and this is why $\mathsf{min}(\mathsf{a})$ remains false in N'. This reveals how the last three lines of the complete translation as given in Figure 3 invalidate N' as a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$.

Correctness of the Translation Function $\mathrm{Tr}_{\mathrm{AT}}$

Let us then address the correctness of the translation function $\mathrm{Tr}_{\mathrm{AT}}$. In order to describe the correspondence between stable models, the following definitions make explicit how a stable model M of a normal program P can be extended to a stable model N of the translation $\mathrm{Tr}_{\mathrm{AT}}(P)$. This is because $\mathrm{Tr}_{\mathrm{AT}}(P)$ involves many new atoms, the truth values of which have to be determined. First of all, we deal with atoms that are essentially defined by the rules of $\mathrm{Tr}_{\mathrm{SUPP}}(P)$ and define the respective extension operator $\mathrm{Ext}_{\mathrm{SUPP}}(P, M)$ for P and M below. Recall that in addition to reproducing M, this part of the translation is responsible for defining the complementary atoms $\overline{\mathsf{a}}$, for which $\mathsf{a} \in \mathrm{Hb}(P)$, and the atoms $\mathsf{bt}(r)$ and $\overline{\mathsf{bt}(r)}$, which detect the satisfaction of $\mathrm{body}(r)$ for rules $r \in P$. Out of these atoms, the ones included in the set $\mathrm{Ext}_{\mathrm{SUPP}}(P, M)$ defined below are supposed be true in the corresponding stable model N of $\mathrm{Tr}_{\mathrm{AT}}(P)$.

Definition 6.14 *For a normal program P and an interpretation M of P, define* $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) =$

$$M \cup \{\overline{\mathsf{a}} \mid \mathsf{a} \in \mathrm{Hb}(P) - M\} \cup \{\mathsf{bt}(r) \mid r \in \mathrm{SR}(P, M)\} \cup \{\overline{\mathsf{bt}(r)} \mid r \in P - \mathrm{SR}(P, M)\}. \tag{6.17}$$

By the following definition, we introduce similar extension operators for the other parts of $\mathrm{Tr}_{\mathrm{AT}}(P)$. For instance, the rules in $\mathrm{Tr}_{\mathrm{CTR}}(P)$ are responsible for selecting correct values for the counters whose purpose is to capture the unique level numbering $\#$ w.r.t. M. As a result, the atoms in $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$ ought to be marked true in N. The last two parts of the translation contribute atoms involved in the constraints on the values of the counters, which implement the maximization/minimization principles from Definition 5.2. Again, the respective extension operators $\mathrm{Ext}_{\mathrm{MAX}}$ and $\mathrm{Ext}_{\mathrm{MIN}}$ determine which atoms evaluate to true given P, M, and $\#$.

Definition 6.15 *For a normal program P, an interpretation M of P, and a function $\# : M \cup \mathrm{SR}(P, M) \to \{0, \ldots, 2^{\nabla P} - 1\}$, define the following extension operators:*

$$\begin{aligned}\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = {} & \bigcup_{\mathsf{a} \in M} \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), \#\mathsf{a}) \cup \\ & \bigcup_{\mathsf{a} \in M} \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt}(\mathsf{a}), \#\mathsf{a} + 1 \bmod 2^{\nabla P})] \cup \\ & \bigcup_{r \in \mathrm{SR}(P,M) \text{ and } \mathrm{body}^+(r) = \emptyset} \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), 1) \cup \\ & \bigcup_{r \in \mathrm{SR}(P,M) \text{ and } \mathrm{body}^+(r) \neq \emptyset} \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#r).\end{aligned} \tag{6.18}$$

$$\begin{aligned}(6.19)\quad \mathrm{Ext}_{\mathrm{MAX}}(P,M,\#) = {} & \{\mathsf{max}(r) \mid r \in \mathrm{SR}(P,M) \text{ and } \mathrm{body}^+(r) \neq \emptyset\} \cup \\ & \bigcup_{r\in\mathrm{SR}(P,M) \text{ and } \mathsf{a}\in\mathrm{body}^+(r)} \mathrm{AT}^{\mathrm{lt}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{nxt}(\mathsf{a}), \#\mathsf{a}+1 \bmod 2^{\nabla P}) \cup \\ & \bigcup_{r\in\mathrm{SR}(P,M) \text{ and } \mathsf{a}\in\mathrm{body}^+(r)} \mathrm{AT}^{\mathrm{eq}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{nxt}(\mathsf{a}), \#\mathsf{a}+1 \bmod 2^{\nabla P}).\end{aligned}$$

$$\begin{aligned}(6.20)\quad \mathrm{Ext}_{\mathrm{MIN}}(P,M,\#) = {} & \{\mathsf{min}(\mathsf{a}) \mid \mathsf{a} \in M\} \cup \\ & \bigcup_{r\in\mathrm{SR}(P,M)} \mathrm{AT}^{\mathrm{lt}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{ctr}(\mathrm{head}(r)), \#\mathrm{head}(r)) \cup \\ & \bigcup_{r\in\mathrm{SR}(P,M)} \mathrm{AT}^{\mathrm{eq}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{ctr}(\mathrm{head}(r)), \#\mathrm{head}(r)).\end{aligned}$$

The four extensions operators introduced so far are combined into one extension operator for the whole translation $\mathrm{Tr}_{\mathrm{AT}}(P)$. It should be yet emphasized that the four sets of atoms involved in Definition 6.16 are disjoint.

Definition 6.16 *For a normal program P, an interpretation $M \subseteq \mathrm{Hb}(P)$ of P, and a function $\# : M \cup \mathrm{SR}(P,M) \to \{0,\ldots,2^{\nabla P}-1\}$, define*

$$\begin{aligned}(6.21)\quad \mathrm{Ext}_{\mathrm{AT}}(P,M,\#) = {} & \mathrm{Ext}_{\mathrm{SUPP}}(P,M) \cup \\ & \mathrm{Ext}_{\mathrm{CTR}}(P,M,\#) \cup \mathrm{Ext}_{\mathrm{MAX}}(P,M,\#) \cup \mathrm{Ext}_{\mathrm{MIN}}(P,M,\#).\end{aligned}$$

The correctness of the translation function $\mathrm{Tr}_{\mathrm{AT}}$ is addressed in Propositions 6.17 and 6.19 as well as Theorem 6.20.

Proposition 6.17 *Let P be a normal program. If M is a stable model of P and $\#$ is the corresponding level numbering w.r.t. M, then the interpretation $N = \mathrm{Ext}_{\mathrm{AT}}(P,M,\#)$ is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$ such that $M = N \cap \mathrm{Hb}(P)$.*

Definition 6.18 *Let P be a normal program, $N \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{AT}}(P))$ an interpretation of the translation $\mathrm{Tr}_{\mathrm{AT}}(P)$, and $M = N \cap \mathrm{Hb}(P)$. Define a function $\# : M \cup \mathrm{SR}(P,M) \to \{0,\ldots,2^{\nabla P}-1\}$ by setting*

1. *$\#\mathsf{a} = \mathrm{val}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), N)$ for atoms $\mathsf{a} \in M$, and*
2. *$\#r = \mathrm{val}_{\nabla P}(\mathsf{ctr}(r), N)$ for rules $r \in \mathrm{SR}(P,M)$.*

Proposition 6.19 *Let P be a normal program. If N is a stable model of the translation $\mathrm{Tr}_{\mathrm{AT}}(P)$, then $M = N \cap \mathrm{Hb}(P)$ is a stable model of P and $N = \mathrm{Ext}_{\mathrm{AT}}(P,M,\#)$ where $\#$ is defined as in Definition 6.18.*

Theorem 6.20 $\mathcal{P} \leq_{\mathrm{PF}} \mathcal{A}$.

However, due to the size and intricacy of $\mathrm{Tr}_{\mathrm{AT}}$, the correctness proofs appear separately in Section 9. An important subsidiary notion used in the proofs is the one of *local stability* given in Definition 6.21 below. As established in Theorem 6.22, atomic programs lend themselves to localizing the fixed point condition behind the stable model semantics. Consequently, the proofs for Propositions 6.17 and 6.17 can be devised one part at a time.

Definition 6.21 *An interpretation I is* ***locally stable*** *w.r.t. a normal program P if and only if $I \cap \mathrm{head}(P) = \mathrm{LM}(P^I)$.*

Theorem 6.22 *Let $P_1, \ldots, P_n$ be* ***atomic*** *normal programs such that the sets $\mathrm{head}(P_1), \ldots, \mathrm{head}(P_n)$ form a partition of $\mathrm{head}(P)$ for $P = \bigcup_{i=1}^{n} P_i$.*

For any $S \subseteq \{1, \ldots, n\}$, an interpretation $M \subseteq \mathrm{head}(P)$ is locally stable w.r.t. $P_S = \bigcup_{i \in S} P_i \iff M$ is locally stable w.r.t. P_i for every $i \in S$.

Moreover, an interpretation $M \subseteq \mathrm{Hb}(P)$ is a stable model of $P \iff M \subseteq \mathrm{head}(P)$ and M is locally stable w.r.t. P_i for every $i \in \{1, \ldots, n\}$.

PROOF. Let us pick any set of indices $S \subseteq \{1, \ldots, n\}$. ($\Longrightarrow$) Let $M \subseteq \mathrm{head}(P)$ be locally stable w.r.t. $P_S = \bigcup_{i \in S} P_i$. Thus we have

$$\begin{array}{rcl} M \cap \mathrm{head}(P_S) & = & M \cap \mathrm{head}(\bigcup_{i \in S} P_i) \\ & = & M \cap (\bigcup_{i \in S} \mathrm{head}(P_i)) \\ & = & \mathrm{LM}(P_S^M) \\ & = & \mathrm{LM}((\bigcup_{i \in S} P_i)^M) \\ & = & \mathrm{LM}(\bigcup_{i \in S} P_i^M) \\ & = & \bigcup_{i \in S} \mathrm{LM}(P_i^M) \end{array} \tag{6.22}$$

where the last equality holds by Lemma 4.9, as P_i^M is atomic and positive for each $i \in S$. Then consider any index $i \in S$. Lemma 4.9 implies that $\mathrm{LM}(P_i^M) = \mathrm{head}(P_i^M) \subseteq \mathrm{head}(P_i)$. Moreover, the properties of P imply that $\bigcup_{j \in S} \mathrm{head}(P_j)$ forms a partition of $\mathrm{head}(P_S)$. Thus $M \cap \mathrm{head}(P_i) = \mathrm{LM}(P_i^M)$ follows by intersecting both sides of the equation (6.22) with $\mathrm{head}(P_i)$. To conclude, M is locally stable w.r.t. P_i for every $i \in S$.

($\Longleftarrow$) Suppose that $M \subseteq \mathrm{head}(P)$ is locally stable w.r.t. P_i for every $i \in S$. Thus $M_i = M \cap \mathrm{head}(P_i) = \mathrm{LM}(P_i^M)$ holds for all $i \in S$. Since $\bigcup_{i \in S} \mathrm{head}(P_i)$ forms a partition of $\mathrm{head}(P_S)$, we obtain $M \cap \mathrm{head}(P_S) = \bigcup_{i \in S} M_i = \bigcup_{i \in S} \mathrm{LM}(P_i^M)$. Since P_i^M is atomic and positive for each $i \in S$, we obtain by Lemma 4.9 that

$$\begin{array}{rcl} M \cap \mathrm{head}(P_S) & = & \bigcup_{i \in S} \mathrm{LM}(P_i^M) \\ & = & \mathrm{LM}(\bigcup_{i \in S} P_i^M) \\ & = & \mathrm{LM}((\bigcup_{i \in S} P_i)^M) \\ & = & \mathrm{LM}(P_S^M). \end{array}$$

Thus M is locally stable w.r.t. P_S.

For the last claim, we note that $M \subseteq \mathrm{Hb}(P)$ is a stable model of P, i.e. $M = \mathrm{LM}(P^M) \iff M \subseteq \mathrm{head}(P)$ and M is locally stable w.r.t. P. □

In analogy to the classes of positive programs, the four classes of normal programs reside in the same expressiveness class if measured by the existence of polynomial and faithful translation function, i.e. the relation $\leq_{\mathrm{FM}}$.

Corollary 6.23 $\mathcal{A} =_{\mathrm{PF}} \mathcal{U} =_{\mathrm{PF}} \mathcal{B} =_{\mathrm{PF}} \mathcal{P}$.

On the other hand, let us consider any class $\mathcal{C} \in \{\mathcal{A}, \mathcal{U}, \mathcal{B}, \mathcal{P}\}$. It follows by Proposition 3.18 that $\mathcal{C}^+ \leq_{\mathrm{PF}} \mathcal{C}$ and by Theorem 4.23 that $\mathcal{C} \not\leq_{\mathrm{PF}} \mathcal{C}^+$. Thus $\mathcal{C}^+ <_{\mathrm{PF}} \mathcal{C}$ holds for all representatives of the two expressiveness classes. The resulting hierarchy of classes of logic programs is illustrated in Figure 2.

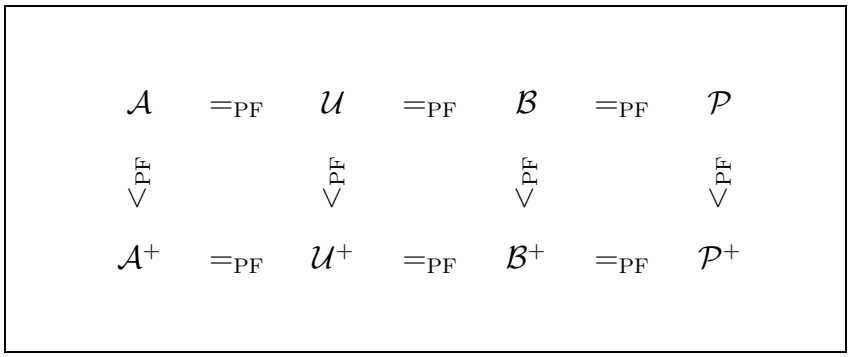

Figure 2: Expressive Power Hierarchy based on polynomial and faithful (PF) translation functions

6.3 Propositional Logic Revisited

In general, it is very challenging to translate a normal program P into a set of clauses so that a one-to-one correspondence of models is obtained. For instance, the approach [5] by Ben-Eliyahu and Dechter is based on a transformation that is clearly polynomial, but the produced set of clauses may possess multiple models corresponding to one stable model of P. However, atomic programs provide a promising intermediary representation that is relatively straightforward to translate into a set of propositional clauses. Here we can apply Clark's program completion as established by Fages [14], but new atoms have to be introduced by the translation function $\mathrm{Tr}_{\mathrm{CL}}$ in order to keep the translation function linear; or even polynomial in the first place.

Definition 6.24 *For an atomic normal program $P \in \mathcal{A}$ and an atom $\mathsf{a} \in \mathrm{Hb}(P)$, let $\mathrm{Def}_P(\mathsf{a}) = \{r \in P \mid \mathrm{head}(r) = \mathsf{a}\}$ and define the set of clauses*

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

where $\mathsf{bt}(r)$ is a new atom for each $r \in P$ and

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

The Herbrand base $\mathrm{Hb}(\mathrm{Tr}_{\mathrm{CL}}(P))$ equals to $\mathrm{Hb}(P) \cup \{\mathsf{bt}(r) \mid r \in P\}$ with $\mathrm{Hb}_{\mathrm{a}}(\mathrm{Tr}_{\mathrm{CL}}(P)) = \emptyset$. The visible part $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{CL}}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$.

The intuitive reading of $\mathsf{bt}(r)$ is the same as in Section 6.2, i.e. $\mathsf{bt}(r)$ is supposed to be true whenever the body of the rule r is true. Roughly speaking, the clauses in the translation ensure that every atom $\mathsf{a} \in \mathrm{Hb}(P)$ is logically equivalent to the disjunction of all bodies of rules $r \in P$ with $\mathrm{head}(r) = \mathsf{a}$. More precisely, clauses of the first two kinds in $\mathrm{Tr}_{\mathrm{CL}}(\mathsf{a}, P)$ enforce the equivalence of each $\mathsf{a} \in \mathrm{Hb}(P)$ with the disjunction $\bigvee\{\mathsf{bt}(r) \mid r \in \mathrm{Def}_P(\mathsf{a})\}$.

On the other hand, each disjunct $\mathsf{bt}(r)$ is made equivalent to the conjunction of negative (classical) literals $\bigwedge\{\neg\mathsf{c} \mid \mathsf{c} \in \mathrm{body}^-(P)\}$ by clauses of the last two kinds in $\mathrm{Tr}_{\mathrm{CL}}(\mathsf{a}, P)$. The net effect is Clark's completion for each $\mathsf{a} \in \mathrm{Hb}(P)$. This leads to a tight correspondence of models as described next.

Definition 6.25 *Given an interpretation $I \subseteq \mathrm{Hb}(P)$ of $P \in \mathcal{A}$, define*

$$\mathrm{Ext}_{\mathrm{CL}}(P, I) = I \cup \{\mathsf{bt}(r) \mid r \in \mathrm{SR}(P, I)\}.$$

Proposition 6.26 *Let P be an atomic normal program. If $M \subseteq \mathrm{Hb}(P)$ is a supported model of P, then $N = \mathrm{Ext}_{\mathrm{CL}}(P, M)$ is a model of $\mathrm{Tr}_{\mathrm{CL}}(P)$ such that $M = N \cap \mathrm{Hb}(P)$.*

PROOF. Let M be a supported model of P and let us assume that $N \not\models \mathrm{Tr}_{\mathrm{CL}}(P)$ holds for the interpretation $N = \mathrm{Ext}_{\mathrm{CL}}(P, M)$. Four cases arise.

1. Suppose that $N \not\models \{\mathsf{a}, \neg\mathsf{bt}(r)\}$ for some $\mathsf{a} \in \mathrm{Hb}(P)$ and $r \in P$ such that $\mathrm{head}(r) = \mathsf{a}$. The truth definition of clauses implies $\mathsf{a} \notin N$ and $\mathsf{bt}(r) \in N$. Then $\mathsf{a} \notin M$ and $r \in \mathrm{SR}(P, M)$ follow by the definition of N. Thus $M \models \mathrm{body}(r)$ and $M \not\models \mathrm{head}(r)$, i.e. $M \not\models r$, a contradiction.

2. Assume that $N \not\models \{\neg\mathsf{a}\} \cup \{\mathsf{bt}(r) \mid r \in \mathrm{Def}_P(\mathsf{a})\}$ for some $\mathsf{a} \in \mathrm{Hb}(P)$. It follows that $\mathsf{a} \in N$ and $\mathsf{bt}(r) \notin N$ for each $r \in \mathrm{Def}_P(\mathsf{a})$. Then the definition of N implies that $\mathsf{a} \in M$ and $r \notin \mathrm{SR}(P, M)$ for each $r \in P$ with $\mathrm{head}(r) = \mathsf{a}$. A contradiction, as M is a supported model of P.

3. Consider the case that $N \not\models \{\mathsf{bt}(r)\} \cup \mathrm{body}^-(r)$ for some $r \in P$. It follows that $\mathsf{bt}(r) \notin N$ and $\mathsf{c} \notin N$ for each $\mathsf{c} \in \mathrm{body}^-(r)$. The interconnection of N and M implies $r \notin \mathrm{SR}(P, M)$ and $\mathsf{c} \notin M$ for each $\mathsf{c} \in \mathrm{body}^-(r)$. Since r is atomic, we obtain $M \models \mathrm{body}(r)$, i.e. $r \in \mathrm{SR}(P, M)$, a contradiction.

4. Suppose that $N \not\models \{\neg\mathsf{bt}(r), \neg\mathsf{c}\}$ for some $r \in P$ and $\mathsf{c} \in \mathrm{body}^-(r)$. Then $\mathsf{bt}(r) \in N$ and $\mathsf{c} \in N$ hold by the truth definition. The definition of N implies $r \in \mathrm{SR}(P, M)$ and $\mathsf{c} \in M$. But then $M \not\models {\sim}\mathsf{c}$ holds and we have $M \not\models \mathrm{body}(r)$ as well as $r \notin \mathrm{SR}(P, M)$, a contradiction.

Hence our assumption must be wrong, i.e. $N \models \mathrm{Tr}_{\mathrm{CL}}(P)$ must hold. □

Proposition 6.27 *Let P be an atomic normal program. If an interpretation $N \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{CL}}(P))$ is a (classical) model $\mathrm{Tr}_{\mathrm{CL}}(P)$, then $M = N \cap \mathrm{Hb}(P)$ is a supported model of P such that $N = \mathrm{Ext}_{\mathrm{CL}}(P, M)$.*

PROOF. Let N be a model of $\mathrm{Tr}_{\mathrm{CL}}(P)$. Let us first establish that $N = \mathrm{Ext}_{\mathrm{CL}}(P, M)$ holds for $M = N \cap \mathrm{Hb}(P)$. This boils down to establishing that $\mathsf{bt}(r) \in N \iff r \in \mathrm{SR}(P, M)$ holds for all $r \in P$. So let us consider any rule $r \in P$. ($\Longrightarrow$) Assume that $\mathsf{bt}(r) \in N$, i.e. $N \models \mathsf{bt}(r)$. Since $N \models \mathrm{Tr}_{\mathrm{CL}}(P)$, we have $N \models \{\neg\mathsf{bt}(r), \neg\mathsf{c}\}$ for any $\mathsf{c} \in \mathrm{body}^-(r)$. It follows that $N \models \neg\mathsf{c}$, i.e. $\mathsf{c} \notin N$. Since $\mathsf{c} \in \mathrm{Hb}(P)$, we obtain $\mathsf{c} \notin M$. Thus $M \models {\sim}\mathsf{c}$ for each $\mathsf{c} \in \mathrm{body}^-(r)$. This implies $M \models \mathrm{body}(r)$, as r is atomic, and $r \in \mathrm{SR}(P, M)$. ($\Longleftarrow$) Suppose that $\mathsf{bt}(r) \notin N$. That is, $N \not\models \mathsf{bt}(r)$. Since $N \models \mathrm{Tr}_{\mathrm{CL}}(P)$, we know that $N \models \bigvee \mathrm{body}^-(r)$. Then there is an

atom $\mathsf{c} \in \mathrm{body}^-(r)$ such that $N \models \mathsf{c}$. Then the definition of M implies $\mathsf{c} \in M$, as $\mathsf{c} \in \mathrm{Hb}(P)$. But then $M \not\models \sim\mathsf{c}$ so that $M \not\models \mathrm{body}(r)$. Thus $r \notin \mathrm{SR}(P, M)$.

Let us then assume that $M \not\models P$. Then there is a rule $r \in P$ such that $M \not\models r$, i.e. $M \models \mathrm{body}(r)$ and $M \not\models \mathrm{head}(r)$. It follows that $r \in \mathrm{SR}(P, M)$ and $\mathrm{head}(r) = \mathsf{a} \notin M$. Since $N = \mathrm{Ext}_{\mathrm{CL}}(P, M)$, we obtain $\mathsf{bt}(r) \in N$ and $\mathsf{a} \notin N$. This implies $N \not\models \{\mathsf{a}, \neg\mathsf{bt}(r)\}$ by the truth definition of clauses. Hence $N \not\models \mathrm{Tr}_{\mathrm{CL}}(P)$, a contradiction.

Finally, we suppose that M is not a supported model, i.e. there is an atom $\mathsf{a} \in M$ such that $r \notin \mathrm{SR}(P, M)$ for all $r \in P$ such that $\mathrm{head}(r) = \mathsf{a}$. Since $\mathsf{a} \in \mathrm{Hb}(P)$, we have $\mathsf{a} \in N$, too. In addition, the fact that $N = \mathrm{Ext}_{\mathrm{CL}}(P, M)$ implies $\mathsf{bt}(r) \notin N$ for each $r \in \mathrm{Def}_P(\mathsf{a})$. It follows that $N \not\models \{\neg\mathsf{a}\} \cup \{\mathsf{bt}(r) \mid r \in \mathrm{Def}_P(\mathsf{a})\}$. Therefore, we have $N \not\models \mathrm{Tr}_{\mathrm{CL}}(P)$, a contradiction. Hence, M is necessarily a (supported) model of P. □

Example 6.28 A logic program P consisting of two rules $r_1 = \mathsf{a} \leftarrow \sim\mathsf{a}$ and $r_2 = \mathsf{a} \leftarrow \sim\mathsf{b}$ has unique stable model $M = \{\mathsf{a}\}$. On the other hand,

$$\begin{aligned}\mathrm{Tr}_{\mathrm{CL}}(P) = \{\ & \mathsf{a} \vee \neg\mathsf{bt}(r_1), \mathsf{a} \vee \neg\mathsf{bt}(r_2), \neg\mathsf{a} \vee \mathsf{bt}(r_1) \vee \mathsf{bt}(r_2), \neg\mathsf{b}, \\ & \mathsf{bt}(r_1) \vee \mathsf{a}, \neg\mathsf{bt}(r_1) \vee \neg\mathsf{a}, \mathsf{bt}(r_2) \vee \mathsf{b}, \neg\mathsf{bt}(r_2) \vee \neg\mathsf{b}\ \}\end{aligned}$$

has a unique model $N = \{\mathsf{a}, \mathsf{bt}(r_2)\}$, as interpretations are restricted to $\mathrm{Hb}(\mathrm{Tr}_{\mathrm{CL}}(P)) = \{\mathsf{a}, \mathsf{b}, \mathsf{bt}(r_1), \mathsf{bt}(r_2)\}$.

The translation function $\mathrm{Tr}_{\mathrm{CL}}$ is clearly non-modular, since the clauses of the type $\{\neg\mathsf{a}\} \cup \{\mathsf{bt}(r) \mid r \in \mathrm{Def}_P(\mathsf{a})\}$ create a dependency between rules possessing the same head a. Let us then address polynomiality and faithfulness as suggested by the one-to-one correspondence obtained in Example 6.28.

Proposition 6.29 *Let P be an atomic normal program. Then $M \subseteq \mathrm{Hb}(P)$ is a stable model of P if and only if M is a supported model of P.*

PROOF. ($\Longrightarrow$) This holds by Proposition 5.7 despite the fact that P is atomic. ($\Longleftarrow$) Let M be a supported model of P. Let us define a function $\#$ from $M \cup \mathrm{SR}(P, M)$ to $\mathbb{N}$ such that $\#\mathsf{a} = 1$ for all $\mathsf{a} \in M$ and $\#r = 1$ for all $r \in \mathrm{SR}(P, M)$. Since P is atomic, we have $\mathrm{body}^+(r) = \emptyset$ for every $r \in P$ and it is easy to inspect from Definition 5.2 that $\#$ is a level numbering w.r.t. M. Thus M is a stable model of P by Theorem 5.8. □

Theorem 6.30 $\mathcal{A} \leq_{\mathrm{PF}} \mathcal{SC}$.

PROOF. A atomic rule $r = \mathsf{a} \leftarrow \sim\mathsf{c}_1, \ldots, \sim\mathsf{c}_m$ consists of $3m + 2$ symbols if each atom counts as one symbol and one symbol is reserved for separating it from other rules. The translation function $\mathrm{Tr}_{\mathrm{CL}}$ translates r effectively into

$$\{\mathsf{a}, \neg\mathsf{bt}(r)\}, \{\mathsf{bt}(r), \mathsf{c}_1, \ldots, \mathsf{c}_m\}, \quad \{\neg\mathsf{bt}(r), \neg\mathsf{c}_1\}, \ldots, \text{ and } \{\neg\mathsf{bt}(r), \neg\mathsf{c}_m\}, \tag{6.23}$$

which contain $10m + 11$ symbols — including separating commas. In addition, the rule r contributes one literal to $\{\neg\mathsf{a}\} \cup \{\mathsf{bt}(r') \mid r' \in \mathrm{Def}_P(\mathsf{a})\}$ which produces two additional symbols for r and 4 symbols for each $\mathsf{a} \in$

$\mathrm{Hb}(P)$. Thus an atomic program P consisting of $\sum_{r\in P}(3 \times |\mathrm{body}^-(r)| + 2)$ symbols is translated into $\mathrm{Tr}_{\mathrm{CL}}(P)$ consisting of $\sum_{r\in P}(10 \times |\mathrm{body}^-(r)| + 13) + 4 \times |\mathrm{Hb}(P)|$ symbols. The translation $\mathrm{Tr}_{\mathrm{CL}}(P)$ can be produced by going through the rules of P, creating the clauses in (6.23), and keeping an account of atoms that appear as heads in the rules. The clause $\{\neg\mathsf{a}\} \cup \{\mathsf{bt}(r') \mid r' \in \mathrm{Def}_P(\mathsf{a})\}$ needs to be created for such atoms. Thus we conclude $\mathrm{Tr}_{\mathrm{CL}}$ to be polynomial.

Then we need to establish the faithfulness of $\mathrm{Tr}_{\mathrm{CL}}$. Let P be an atomic normal program. Note that $\mathrm{Hb}(P) \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{CL}}(P))$ and $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{CL}}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$ hold directly by Definition 6.24. It follows by Propositions 6.29 and 6.26 that there is an extension function $\mathrm{Ext}_{\mathrm{CL}} : \mathrm{SM}(P) \to \mathrm{CM}(\mathrm{Tr}_{\mathrm{CL}}(P))$ that maps $M \in \mathrm{SM}(P)$ into $N = \mathrm{Ext}_{\mathrm{CL}}(P, M)$ included in $\mathrm{CM}(\mathrm{Tr}_{\mathrm{CL}}(P))$ such that $M = N \cap \mathrm{Hb}(P)$. Moreover, Propositions 6.27 and 6.29 imply that that if $N \in \mathrm{CM}(\mathrm{Tr}_{\mathrm{CL}}(P))$, then $M = N \cap \mathrm{Hb}(P) \in \mathrm{SM}(P)$ and $N = \mathrm{Ext}_{\mathrm{CL}}(P, M)$. Thus we may conclude $\mathrm{Tr}_{\mathrm{CL}}$ to be faithful by Theorem 3.19. □

Corollary 6.31 $\mathcal{SC} =_{\mathrm{PF}} \mathcal{A} =_{\mathrm{PF}} \mathcal{U} =_{\mathrm{PF}} \mathcal{B} =_{\mathrm{PF}} \mathcal{P}$.

Corollary 6.32 $\mathcal{C}^+ <_{\mathrm{PF}} \mathcal{SC}$ *holds for any* $\mathcal{C} \in \{\mathcal{A}, \mathcal{U}, \mathcal{B}, \mathcal{P}\}$.

```
% TR_SUPP(P)

not_a :- not a.
bt_r :- not not_a. not_bt_r :- not bt_r.
a :- not not_bt_r.

% TR_CTR(P): SEL(ctr_a, not_a), NXT(ctr_a, nxt_a, not_a) and SEL(ctr_r, not_bt_r)

ctr_a_1 :- not not_ctr_a_1, not not_a. not_ctr_a_1 :- not ctr_a_1, not not_a.
ctr_a_2 :- not not_ctr_a_2, not not_a. not_ctr_a_2 :- not ctr_a_2, not not_a.
not_nxt_a_1 :- not not_ctr_a_1, not not_ctr_a_2, not nxt_a_2, not not_a.
nxt_a_1 :- not ctr_a_1, not not_ctr_a_2, not nxt_a_2, not not_a.
not_nxt_a_1 :- not ctr_a_1, not ctr_a_2, not not_a.
nxt_a_1 :- not not_ctr_a_1, not ctr_a_2, not not_a.
not_nxt_a_1 :- not ctr_a_1, not not_nxt_a_2, not not_a.
nxt_a_1 :- not not_ctr_a_1, not not_nxt_a_2, not not_a.
not_nxt_a_2 :- not not_ctr_a_2, not not_a.
nxt_a_2 :- not ctr_a_2, not not_a.
ctr_r_1 :- not not_ctr_r_1, not not_a. not_ctr_r_1 :- not ctr_r_1, not not_a.
ctr_r_2 :- not not_ctr_r_2, not not_a. not_ctr_r_2 :- not ctr_r_2, not not_a.

% TR_MAX(P): LT(ctr_r, nxt_a, not_bt_r) and EQ(ctr_r, nxt_a, not_bt_r) plus 3 rules

lt_ctr_r_nxt_a_1 :- not ctr_r_1, not not_nxt_a_1, not not_bt_r.
lt_ctr_r_nxt_a_2 :- not ctr_r_2, not not_nxt_a_2, not not_bt_r.
lt_ctr_r_nxt_a_1 :-
  not ctr_r_1, not nxt_a_1, not not_lt_ctr_r_nxt_a_2, not not_bt_r.
lt_ctr_r_nxt_a_1 :-
  not not_ctr_r_1, not not_nxt_a_1, not not_lt_ctr_r_nxt_a_2, not not_bt_r.
not_lt_ctr_r_nxt_a_1 :- not lt_ctr_r_nxt_a_1, not not_bt_r.
not_lt_ctr_r_nxt_a_2 :- not lt_ctr_r_nxt_a_2, not not_bt_r.
not_eq_ctr_r_nxt_a :- not ctr_r_1, not not_nxt_a_1, not not_bt_r.
not_eq_ctr_r_nxt_a :- not not_ctr_r_1, not nxt_a_1, not not_bt_r.
not_eq_ctr_r_nxt_a :- not ctr_r_2, not not_nxt_a_2, not not_bt_r.
not_eq_ctr_r_nxt_a :- not not_ctr_r_2, not nxt_a_2, not not_bt_r.
eq_ctr_r_nxt_a :- not not_eq_ctr_r_nxt_a, not not_bt_r.
x :- not x, not not_bt_r, not not_lt_ctr_r_nxt_a_1.
max_r :- not not_bt_r, not not_eq_ctr_r_nxt_a.
x :- not x, not not_bt_r, not max_r.

% TR_MAX(P): LT(ctr_r, ctr_a, not_bt_r) and EQ(ctr_r, ctr_a, not_bt_r) plus 3 rules

lt_ctr_r_ctr_a_1 :- not ctr_r_1, not not_ctr_a_1, not not_bt_r.
lt_ctr_r_ctr_a_2 :- not ctr_r_2, not not_ctr_a_2, not not_bt_r.
lt_ctr_r_ctr_a_1 :-
  not ctr_r_1, not ctr_a_1, not not_lt_ctr_r_ctr_a_2, not not_bt_r.
lt_ctr_r_ctr_a_1 :-
  not not_ctr_r_1, not not_ctr_a_1, not not_lt_ctr_r_ctr_a_2, not not_bt_r.
not_lt_ctr_r_ctr_a_1 :- not lt_ctr_r_ctr_a_1, not not_bt_r.
not_lt_ctr_r_ctr_a_2 :- not lt_ctr_r_ctr_a_2, not not_bt_r.
not_eq_ctr_r_ctr_a :- not ctr_r_1, not not_ctr_a_1, not not_bt_r.
not_eq_ctr_r_ctr_a :- not not_ctr_r_1, not ctr_a_1, not not_bt_r.
not_eq_ctr_r_ctr_a :- not ctr_r_2, not not_ctr_a_2, not not_bt_r.
not_eq_ctr_r_ctr_a :- not not_ctr_r_2, not ctr_a_2, not not_bt_r.
eq_ctr_r_ctr_a :- not not_eq_ctr_r_ctr_a, not not_bt_r.
y :- not y, not not_bt_r, not not_lt_ctr_r_ctr_a_1.
min_a :- not not_bt_r, not not_eq_ctr_r_ctr_a.
y :- not y, not not_a, not min_a.
```

Figure 3: A translation of a normal program into atomic one

7 RELATED WORK

Let us first comment on the major changes to an earlier published draft of this report [24] in which the systematic classification method for logic programs, as presented in Section 3.4, was initiated. In contrast to [24], slightly different requirements on translation functions are currently imposed in Section 3.3.

1. Firstly, the notion of modularity is now more fine-grained due to module conditions M1–M4 introduced in Definition 3.15. That is, (3.6) is supposed to hold in limited context while P and Q can be arbitrary according to [24]. Moreover, our earlier approach assumes that $\mathrm{Tr}(P) = P$ for all $P \in \mathcal{C}_1$ if $\mathrm{Tr} : \mathcal{C}_1 \to \mathcal{C}_2$ and $\mathcal{C}_1 \subset \mathcal{C}_2$, i.e. $\mathcal{C}_1$ has a more restricted syntax than $\mathcal{C}_2$. The net effect is that

   $$\mathrm{Tr}(P \cup Q) = P \cup \mathrm{Tr}(Q) \tag{7.1}$$

   should hold for all $P \in \mathcal{C}_1$ and $Q \in \mathcal{C}_2$ whenever $\mathcal{C}_1 \subset \mathcal{C}_2$, if Tr is to be modular. This notion produces analogous intranslatability results given a chain of classes of logic programs — like $\mathcal{A} \subset \mathcal{U} \subset \mathcal{B} \subset \mathcal{P}$ in [24] and this report. However, problems arise if classes, which are syntactically different, such as $\mathcal{SC}$ distinguished in Section 3.1, are taken into consideration. Then it is not guaranteed that the composition of modular translations is modular (c.f. Proposition 3.17). Moreover, it is not clear how to interpret (7.1) if P and $\mathrm{Tr}(Q)$ belong to syntactically different classes of logic programs. For these reasons, we employ a weaker notion of modularity in this report. As a consequence, the intranslatability results obtained in Section 4 become stronger than those established in [24].

2. A further difference concerns the notion of faithfulness. In [24], a faithful translation function $\mathrm{Tr} : \mathcal{C}_1 \to \mathcal{C}_2$ must satisfy $\mathrm{Hb}(P) \subseteq \mathrm{Hb}(\mathrm{Tr}(P))$ and there must exist a bijection $f : \mathrm{Sem}_{\mathcal{C}_1}(P) \to \mathrm{Sem}_{\mathcal{C}_2}(\mathrm{Tr}(P))$ such that $M = f(M) \cap \mathrm{Hb}(P)$ for all $M \in \mathrm{Sem}_{\mathcal{C}_1}(P)$. Compared to this, we employ a weaker notion of faithfulness in this report. This is because the notion of faithfulness is restricted to visible atoms only, and it is enough that the semantics is preserved up to $\mathrm{Hb}_{\mathrm{v}}(P) \subseteq \mathrm{Hb}(P)$. That is, $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P))$ and $M \cap \mathrm{Hb}_{\mathrm{v}}(P) = f(M) \cap \mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}(P))$ in the bijective relationship described above. Again, the weaker notion of faithfulness is in favor of the intranslatability results, which become stronger in this way.

Otherwise, the resulting classification of logic programs are analogous (Figure 1 vs. [24, Figure 2]), except that the comparison with propositional logic is more natural using the current criteria, because polynomiality, faithfulness and modularity are properly preserved under composition. By designing the relations $\leq_{\mathrm{PFM}}$, $<_{\mathrm{PFM}}$, $=_{\mathrm{PFM}}$, ... based on these properties, we have accommodated the classification method proposed for non-monotonic logics [22, 26] to the case of logic programs. The frameworks are analogous, but somewhat different. Most importantly, the semantics of a *non-monotonic theory* is determined by a set of *extensions* which are typically propositionally closed theories[6]. In contrast to this, we assume that the semantics of a logic

[6] Recall that a propositionally closed theory is fully determined by the set of its models.

program P is determined by a set of interpretations/models. The notions of modularity are also different due to major syntactic differences.

Let us then briefly comment on computational complexity. As established by Marek and Truszczyński [34], the problem of checking whether a normal logic program P has a stable model forms an NP-complete decision problem. The translation function $\mathrm{Tr}_{\mathrm{AT}}$ presented in Section 4.4 implies that the satisfiability problem SAT is polynomial time reducible to the problem of checking whether an *atomic/unary/binary* normal logic program has a stable model. This indicates that the computational complexity of the latter problem remains NP-complete under the three syntactic restrictions used in this report. This indicates that the expressive powers of the classes $\mathcal{A}$, $\mathcal{U}$, and $\mathcal{B}$ cannot be differentiated in terms of traditional complexity measures. In contrast to this, the relation $<_{\mathrm{PFM}}$ based on the existence of a polynomial, faithful and modular translation function enables us to detect strict differences. This is mainly because the reducibilities involved in complexity results preserve only the yes/no answers to decision problems. Otherwise transformations between decision problems can be arbitrary as long as they remain computable in polynomial time. As it is clear by Theorems 4.7, 4.10, 4.18, 4.21, 4.23, and 4.29 presented in Section 4 faithfulness and modularity play a central role in our intranslatability results.

As shown in Section 4.4, normal programs cannot be translated into sets of clauses in a faithful and modular way. Niemelä [36, Proposition 4.3] provides a formal counter-example in this respect, too, but the result is based on quite different notions of faithfulness and modularity: the existence of models is to be preserved, and Tr is considered modular if $P \cup F$ and $\mathrm{Tr}(P) \cup F$ are equally satisfiable where F is a very simple program, namely a set of facts; or a positive atomic program in our terminology. Of course, the aim is to make that particular intranslatability result as strong as possible. Despite this particular intranslatability result, the composition of the translation functions $\mathrm{Tr}_{\mathrm{AT}}$ and $\mathrm{Tr}_{\mathrm{CL}}$ from Section 6 is sufficient to reduce normal logic programs into propositional satisfiability. The resulting translation function is definitely not modular, but still highly structural so that actual translations can be computed in a quite systematic fashion. On the other hand, Niemelä [36] presents the basic technique to express propositional satisfiability problems as normal logic programs — the objective being to preserve the existence of models. However, the translation function $\mathrm{Tr}_{\mathrm{LP}}$ from Section 4.4 is designed to meet stronger criteria. First, a one-to-one correspondence of models is established. Second, particular attention is paid to make $\mathrm{Tr}_{\mathrm{LP}}$ modular so that clauses can be translated into rules on clause-by-clause basis.

Partial evaluation techniques have been introduced to unfold rules of programs in a semantics preserving way. A good example in this respect is an approach by Brass and Dix [6]. They propose *equivalence transformations* for normal and *disjunctive* logic programs under the stable model semantics [18]. Let us describe these transformations in case of normal logic programs. Two of their transformations eliminate *tautological and inapplicable* rules, which are rules (2.1) such that $\mathsf{a} = \mathsf{a}_i$ for some i, and rules (2.1) such that $\mathsf{a}_i = \mathsf{b}_j$ for some i and j, respectively. The third transformation is particularly interesting from the perspective of this report, as it affects the number of positive body literals. When a rule (2.1) in a normal program P is *partially*

evaluated with respect to one of its positive body literals a_i, it is replaced by a rule

$$\mathsf{a} \leftarrow \mathsf{a}_1, \ldots, \mathsf{a}_{i-1}, \mathsf{c}_1, \ldots, \mathsf{c}_k, \mathsf{a}_{i+1}, \ldots, \mathsf{a}_n, {\sim}\mathsf{b}_1, \ldots, {\sim}\mathsf{b}_m, {\sim}\mathsf{d}_1, \ldots, {\sim}\mathsf{d}_l$$

for every rule $\mathsf{a}_i \leftarrow \mathsf{c}_1, \ldots, \mathsf{c}_k, {\sim}\mathsf{d}_1, \ldots, {\sim}\mathsf{d}_l$ of P having a_i as the head. In this way, the positive occurrences of a_i are replaced by each rule defining a_i. Thus, partial evaluation may have a quite opposite effect compared to the goals of this paper, as it might increase the number of positive body literals. It is quite easy to see that the translation function $\mathrm{Tr}_{\mathrm{PE}}$ corresponding to partial evaluation is not modular. It is also possible to construct examples for which $\mathrm{Tr}_{\mathrm{PE}}$ causes an exponential blow-up in the length of the program.

Antoniu et al. [1] apply a modularity condition when developing normal forms for Nute's *defeasible logic* [37]. Although defeasible logic is based on a completely different semantics, its rule-based syntax makes it reminiscent of normal programs. Thus a comparison is called for. Firstly, Antoniu et al. consider a translation function Tr to be *correct*, if $D \equiv_{L(D)} \mathrm{Tr}(D)$ for every D. Here $\equiv$ denotes semantical equivalence, i.e., the theories yield exactly the same conclusions in the language $L(D)$ of D. This is somewhat analogous to the notion of faithfulness employed in this paper, but certain differences remain. The first is due to the proof-theoretic semantics of defeasible which assigns a unique set of conclusions to each theory. The second is that our notion is less constrained, since the preservation of conclusions is restricted to visible atoms only. A further property addressed in [1] is *incrementality*, which presumes that the translation function Tr satisfies $D_1 \cup D_2 \equiv_{L(D_1)\cup L(D_2)} \mathrm{Tr}(D_1) \cup \mathrm{Tr}(D_2)$ for every D_1 and D_2. In the presence of correctness, this equation implies $D_1 \cup D_2 \equiv_{L(D_1)\cup L(D_2)} \mathrm{Tr}(D_1 \cup D_2)$, the form of which is very close to the equation (3.6) involved in our definition of modularity. The main difference is that our definition of modularity is based on syntactical equality $=$ rather than semantical equivalence $\equiv_{\mathrm{v}}$. Moreover, there is no counterpart to module conditions in the approach by Antoniu et al. Actually, Antoniu et al. reserve the term *modularity* for a stronger property. For this, the translation function Tr must satisfy $D_1 \cup D_2 \equiv_{L(D_1)\cup L(D_2)} D_1 \cup \mathrm{Tr}(D_2)$ for any defeasible theories D_1 and D_2.[7] If Tr is correct, then modularity is implied by $\mathrm{Tr}(D_1 \cup D_2) \equiv_{L(D_1)\cup L(D_2)} D_1 \cup \mathrm{Tr}(D_2)$ — an obvious analog of (7.1) which makes it impossible to apply such a notion to translation functions between classes of logic programs that are syntactically different.

Ben-Eliyahu and Dechter [5] study the possibilities of reducing *head-cycle-free* disjunctive logic programs, under the stable model semantics [18], to propositional logic. Since normal programs are special cases of head-cycle-free disjunctive programs, a comparison with our results follows. One of the results obtained by Ben-Eliyahu and Dechter [5, Theorem 2.8] is a characterization of stable models that resembles the one developed in Section 5. However, they impose weaker conditions on level numberings. That is, they insist on the existence of a function $f : \mathrm{Hb}(P) \to \mathbb{N}^+$ such that for each $\mathsf{a} \in M$, there is a rule $r \in \mathrm{SR}(P, M)$ satisfying $f(\mathsf{b}) < f(\mathsf{a})$ for every $\mathsf{b} \in \mathrm{body}^+(r)$. It is easy to see that a level numbering $\#$ conforming to

[7] Thus any modular transformation is also incremental and correct [1].

Definition 5.2 can be extended to such a function f, but such functions are by no means unique even if the range of f is limited. This is in contrast to Theorem 5.8 where the uniqueness of level numberings is established. The translation function $\mathrm{Tr_{BD}}$ (called *translate-2* in [5]) produces a propositional theory $\mathrm{Tr_{BD}}(P)$ that consists of four parts. The first two parts ensure that each model N of $\mathrm{Tr_{BD}}(P)$ captures a classical model M of P. The third part makes M a supported and stable model of P whereas the fourth part can be neglected in case of normal programs. In particular, the fact that $f(\mathsf{a}) = i$ holds for an atom $\mathsf{a} \in \mathrm{Hb}(P)$ is expressed by making a new atom $\mathsf{in}(\mathsf{a})_i$ true in N. In our approach, similar objectives can be identified for the parts of $\mathrm{Tr_{AT}}(P)$ given in Definition 6.12. In contrast to the composition $\mathrm{Tr_{AT}} \circ \mathrm{Tr_{CL}}$, the translation function $\mathrm{Tr_{BD}}$ does not necessarily yield a one-to-one correspondence between the stable models of P and the classical models of the translation. This is because the level numberings used by Ben-Eliyahu and Dechter are not unique. Moreover, the language of P is not preserved by $\mathrm{Tr_{BD}}$, i.e., $\mathrm{Hb}(P) \cap \mathrm{Hb}(\mathrm{Tr_{BD}}(P)) = \emptyset$. Thus $\mathrm{Tr_{BD}}$ is far from being faithful in the sense given by Definition 3.14. A further difference is that $||\mathrm{Tr_{BD}}(P)||$ is quadratic in $||P||$ in the worst case. The translation functions developed in this report are more compact: $||\mathrm{Tr_{CL}}(\mathrm{Tr_{AT}}(P))||$ is of order $||P|| \times \nabla P$, as a binary encoding of level numbers is used.

There are also other characterizations of stable models that are closely related to the one established in Section 5. Fages [15] calls an interpretation $I \subseteq \mathrm{Hb}(P)$ of a normal program P *well-supported* if and only if there exists a strict well-founded partial order $\prec$ on I such that for any atom $\mathsf{a} \in I$, there exists $r \in \mathrm{SR}(P, I)$ satisfying $\mathrm{head}(r) = \mathsf{a}$ and $\mathsf{b} \prec \mathsf{a}$ for all $\mathsf{b} \in \mathrm{body}^+(r)$. The basic result [15, Theorem 3.2] that well-supported models of a normal program P are stable models of P, and vice versa. In fact, it is possible to associate such an ordering with a level numbering conforming to Definition 5.2: just define $\mathsf{a} \prec \mathsf{b} \iff \#\mathsf{a} < \#\mathsf{b}$ for any $\mathsf{a} \in I$ and $\mathsf{b} \in I$. The resulting ordering can be considered as a canonical one, as $\#$ is known to be unique by Theorem 5.8. Moreover, Fages distinguishes *positive order consistent* normal programs whose models are necessarily well-supported. As a consequence, the classical models of the completed program P [7], or supported models of P, coincide with the stable models of P. Quite recently, Babovich et al. [4] and also Erdem and Lifschitz [13] generalize Fages' results by introducing the notion of *tightness* for logic programs. The tightness of a logic program P is defined relative a set atoms $A \subseteq \mathrm{Hb}(P)$, which makes Fages' theorem applicable to a wider range of programs. To understand the contribution of this report in this respect, let us point out that atomic normal programs are automatically positive order consistent, or *absolutely* tight in the terminology of [13]. Therefore, arbitrary normal programs can be transformed into absolutely tight ones in a fairly systematic fashion by applying the translation function $\mathrm{Tr_{AT}}$ presented in Section 6. A further implication is that a *transitive closure* of relation can be properly captured with classical models. This is has already been established by Erdem and Lifschitz [12] for relations that can be represented in terms of a tight program.

As already discussed in Section 3.2, a basic notion of equivalence is obtained for a given class of programs $\mathcal{C}$ by requiring that programs to possess the same stable models, i.e. $P \equiv Q \iff \mathrm{Sem}_{\mathcal{C}}(P) = \mathrm{Sem}_{\mathcal{C}}(Q)$. Lifs-

chitz et al. [29] study a stronger condition, which involves an arbitrary context $R \in \mathcal{C}$ in which P and Q could be placed as subprograms. That is, P and Q are *strongly equivalent*, denoted by $P \equiv_{\mathrm{s}} Q \iff$ for all $R \in \mathcal{C}$, $\mathrm{Sem}_{\mathcal{C}}(P \cup R) = \mathrm{Sem}_{\mathcal{C}}(Q \cup R)$. Lifschitz et al. [29] and later Turner [43] characterize strong equivalence in various ways, e.g., using Heyting's logic of here-and-there. We find the equivalence relations $\equiv_{\mathrm{v}}$ and $\equiv_{\mathrm{w}}$, as introduced in Section 3.2, more practically oriented than the two relations above. It is very typical that logic programs contain additional atoms for knowledge representation purposes. Such atoms act as auxiliary concepts which are not directly relevant for the problem being solved and can be hidden from the user. Therefore, it is justifiable to omit such atoms as long as the equivalence of logic programs is concerned. In case of hidden atoms, the relations $\equiv$ and $\equiv_{\mathrm{v}}$ are quite different, but as established in Proposition 3.11 we have $P \equiv Q \iff P \equiv_{\mathrm{v}} Q \iff P \equiv_{\mathrm{w}} Q$ when $\mathrm{Hb}_{\mathrm{v}}(P) = \mathrm{Hb}(P) = \mathrm{Hb}(Q) = \mathrm{Hb}_{\mathrm{v}}(Q)$.

8 CONCLUSIONS

This report concentrates on the problem of reducing the number of positive subgoals in the bodies of rules. To address this problem, we propose a framework based on PFM translation functions in Section 3; the aim being a comparison classes of logic programs — obtained by restricting the number of positive subgoals in rules — on the basis of their expressive power.

Retrospectively speaking, the adjustment of the framework, which was initiated in [24], involved many objectives that had to be settled. For instance, our preliminary comparisons with propositional satisfiability suggested that the framework should be general enough to enable a comparison of a variety of classes of logic programs, which may differ by syntax or semantics — or even both. Furthermore, we realized that the notion of modularity employed in [24] does not generalize properly for the comparison of classes that do not share syntax. Thus extra care was needed in order to guarantee that the requirements on PFM translation functions are preserved under compositions. A further objective was to keep our preliminary (in)translatability results [24] valid under any updates that seemed necessary to the first version of the framework. We believe that the current framework meets these objectives successfully, as indicated by the results established in the rest of this report. However, the development of the underlying theory involved many important technical details such as visibility of atoms, mechanisms to extend Herbrand bases, notions of equivalence, and module conditions. Many of these ideas raised from our practical experiences with answer set programming and existing implementations.

The expressiveness analysis performed in Section 4.2 is quite straightforward, but it reveals the main constituents of *monotonic* rule-based reasoning. In the simplest possible form, we have just sets of atomic rules $\mathsf{a} \leftarrow$ stating that certain atoms are true in the world; and no further inferences are possible. Unary rules enrich this setting by allowing *chained inferences* with rules, e.g. we can infer a using rules $\mathsf{a} \leftarrow \mathsf{b}$; $\mathsf{b} \leftarrow \mathsf{c}$; and $\mathsf{c} \leftarrow$. In the richest form, we have binary rules that incorporate conjunctive conditions to rule-based reasoning. For instance, a follows by the rules $\mathsf{a} \leftarrow \mathsf{b}, \mathsf{c}$; $\mathsf{b} \leftarrow \mathsf{d}$; $\mathsf{c} \leftarrow \mathsf{d}$; and $\mathsf{d} \leftarrow$. Non-binary rules, which have more than two positive subgoals, are easily reducible to these primitive forms. In contrast to this, binary and unary rules are not expressible in a modular way using unary and atomic rules, respectively, as implied by the formal counter examples.

The analysis continues in Section 4.3 and it is established that this setting is not affected even if normal logic programs are considered. That is, negation as failure is insufficient to fully compensate conjunctive conditions nor chained inferences (c.f. Example 4.20). Looking back to the expressive power hierarchy in Figure 1, the number of positive body literals appears to be an essential syntactic restriction, as strict differences in expressive power can be established. Thus, in analogy to our previous experience on classifying non-monotonic logics [22], the method based on PFM translation functions yields a more accurate measure of expressive power than the levels of polynomial time hierarchy (PH) do. It is also interesting to realize that classical sets of clauses do not properly capture reasoning with (atomic) normal programs, as shown in Section 4.4. There is also practical evidence for this,

as many problems are more easily formalized using rules rather than clauses.

The characterization of stable models developed in Section 5 reveals that the computation of the least model for a positive normal program can be understood as a minimization/maximization process. As discussed in Section 7, a particular novelty of a level numbering conforming to Definition 5.2 is that the values assigned to atoms are uniquely determined. This is in sharp contrast with earlier characterizations of stable models, where similar numberings are used to distinguish stable models, but the value assignment can be done even in infinitely many ways. Due to the tight correspondence of models encompassed in the notion of faithfulness employed in this report, unique level numberings are crucial for the main objective of Section 6, i.e. a polynomial and faithful translation of normal programs into atomic ones.

In Section 6, we try to complete our view by considering non-modular alternatives in cases where modular translations turned out to be impossible. As a fundamental result, we develop a counter-based approach for translating normal programs into atomic ones. Compared to earlier attempts, there are several distinctive features in our approach. All finite normal programs can be covered and a bijective relationship of models is obtained. Moreover, the translation function $\mathrm{Tr}_{\mathrm{AT}}$ preserves the Herbrand base of the program, only new atoms are added. The length of the translation $||\mathrm{Tr}_{\mathrm{AT}}(P)||$ as well as the translation time are of order $||P|| \times \log_2 |\mathrm{Hb}(P)|$, indicating that $\mathrm{Tr}_{\mathrm{AT}}$ is sub-quadratic. We consider this as a breakthrough, since the best known transformation to date [5] is quadratic. Nevertheless, it should be noted that the translation function $\mathrm{Tr}_{\mathrm{AT}}$ is far from being optimal. There are several techniques that can be used to decrease the number of rules that have to be generated for a particular normal program; and the number of binary counters as well as the number of bits involved in them. One technique is to compute the *strongly connected components* of P, as already suggested by Ben-Eliyahu and Dechter [5] in case of their translation, and to apply $\mathrm{Tr}_{\mathrm{AT}}$ componentwise. However, we leave such optimizations to be considered elsewhere, as the main interest in the current work is to establish a translation function possessing certain, but still quite promising, properties.

Let us then return to the fundamental question that initiated this work. The formal counter examples crafted in the proofs of Theorems 4.18 and 4.21 indicate that it is impossible to rewrite rules such that all positive body literals are removed, if we expect faithfulness and modularity from such transformations. On the other hand, as established in Theorem 6.20, there are compensatory polynomial and faithful techniques, but which cannot be applied on a rule-by-rule basis. For instance, we have to know ∇P before atoms and rules can be translated. Thus our final answer to the question is affirmative: positive body literals can be removed, but the price may be quite high. That is, lots of extra rules involving new atoms may be needed, as a consequence of which the search for stable models is likely to degrade.

To conclude, binary rules tend to block contrapositive inference in practice. To see this, consider a binary rule $\mathsf{a} \leftarrow \mathsf{b}, \mathsf{c}$; the head of which is is known to be false in a stable model M under construction. Given that the truth values of b and c are not yet known, all we can infer that b is false in M, *or* c is false in M, in order to satisfy the rule in M. Recall that M must be a classical model as well. This leads to a case analysis which can be in

the worst case at least as expensive as ordinary branching with respect to b or c. That is, analyzing separately the cases that b (resp. c) is true in M and b (resp. c) is false in M. Such a setting is illustrated by our final example.

Example 8.1 Consider a binary normal logic program

$$P = \{\mathsf{a} \leftarrow \mathsf{b}, \mathsf{c};\ \ \mathsf{b} \leftarrow \sim\mathsf{b}_1;\ \ \mathsf{b}_1 \leftarrow \sim\mathsf{b};\ \ \mathsf{c} \leftarrow \sim\mathsf{c}_1;\ \ \mathsf{c}_1 \leftarrow \sim\mathsf{c}\}$$

which has four stable models: $M_1 = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, $M_2 = \{\mathsf{b}, \mathsf{c}_1\}$, $M_3 = \{\mathsf{b}_1, \mathsf{c}\}$ and $M_4 = \{\mathsf{b}_1, \mathsf{c}_1\}$. Suppose we would like to compute the stable models M of P in which a is false. As suggested by the contrapositive interpretation of $\mathsf{a} \leftarrow \mathsf{b}, \mathsf{c};$ one possibility is to branch the search using the conditions that (i) b is false in M and (ii) c is false in M. While analyzing the case (i), we find the stable models M_3 and M_4. On the other hand, the stable models M_2 and M_4 are discovered when (ii) is analyzed.[8] Thus M_4 is encountered twice during the search, as it satisfies both assumptions on M. Another approach is to branch according to the condition (i) above and the condition that (iii) b is true in M. In the case (iii), the stable model M_2 is found directly. In the latter setting, each stable model is encountered exactly once, because the assumptions on M are mutually exclusive. □

Nevertheless, we do not claim that contrapositive reasoning is not useful. For instance, if a and b are known to be false and true, respectively, then it is reasonable to infer that c must be false, too, in the setting of Example 8.1. However, depending on our assumptions on the truth values of atoms, binary rules may block contrapositive reasoning in practice.

8.1 Future Work

The current expressive power hierarchy in Figure 1 was obtained as a by-product while analyzing the possibilities for reducing the number of positive body literals. It is clear that the hierarchy can be extended by analyzing the expressiveness of other classes of logic programs. One direction is to consider classes with richer syntax such as *extended programs* [17], *disjunctive programs* [18] and *nested programs* [30]. The other is to take different semantics, such as the well-founded semantics [45], into consideration. Our recent results on partial stable models [27] and disjunctive programs [25] provide a promising starting point in this respect. A comparison with Schlipf's results [40] is also called for in the case of normal programs. We are also developing an optimized implementation of the translation function $\mathrm{Tr}_{\mathrm{AT}}$ presented in Section 6. This work will include comparison with other similar systems such as ASSAT [31], CMODELS [3] and QUIP [10].

8.2 Acknowledgments

The author wishes to thank Mirek Truszczyński for his suggestion to apply techniques from [21] to the research problem addressed in this report as well as anonymous referees of the Computational Logic 2000 conference for

[8] Actually, the analysis above can be performed with SMODELS using compute statements as demonstrated in [24].

comments and suggestions for improvements. This work has been partially supported by the Academy of Finland under projects *"Constraint Programming Based on Default Rules"* (#43963) and *"Applications of Rule-Based Constraint Programming"* (#53695).

REFERENCES

[1] G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, April 2001.

[2] K.R. Apt, H.A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann Publishers, Los Altos, 1988.

[3] Y. Babovich. CMODELS — a tool for computing answer sets using sat solvers. `http://www.cs.utexas.edu/users/tag/cmodels.html`, 2003. Computer Program.

[4] Y. Babovich, E. Erdem, and V. Lifschitz. Fages' theorem and answer set programming. In *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning*, Breckenridge, Colorado, USA, April 2000. cs.AI/0003042.

[5] R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12(1–2):53–87, 1994.

[6] S. Brass and J. Dix. Semantics of (disjunctive) logic programs based on partial evaluation. *Journal of Logic Programming*, 38(3):167–213, 1999.

[7] K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[8] Y. Dimopoulos, B. Nebel, and J. Koehler. Encoding planning problems in non-monotonic logic programs. In *Proceedings of the Fourth European Conference on Planning*, pages 169–181, Toulouse, France, September 1997. Springer-Verlag.

[9] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.

[10] U. Egly, T. Eiter, V. Klotz, H. Tompits, and S. Woltran. Computing stable models with quantified boolean formulas: Some experimental results. In *AAAI Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning*. AAAI, 2001.

[11] Thomas Eiter, Nicola Leone, Cristinel Mateis, Gerald Pfeifer, and Francesco Scarnello. The KR system DLV: Progress report, comparisons and benchmarks. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning*, pages 406–417, Trento, Italy, June 1998. Morgan Kaufmann Publishers.

[12] E. Erdem and V. Lifschitz. Transitive closure, answer sets and predicate completion. In *AAAI Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning*. AAAI, 2001.

[13] E. Erdem and V. Lifschitz. Tight logic programs. *Theory and Practice of Logic Programming*, 3(4–5):499–518, 2003.

[14] F. Fages. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.

[15] F. Fages. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.

[16] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080, Seattle, USA, August 1988. The MIT Press.

[17] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proceedings of the 7th International Conference on Logic Programming*, pages 579–597, Jerusalem, Israel, June 1990. The MIT Press.

[18] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[19] G. Gottlob. Translating default logic into standard autoepistemic logic. *Journal of the Association for Computing Machinery*, 42(2):711–740, 1995.

[20] T. Imielinski. Results on translating defaults to circumscription. *Artificial Intelligence*, 32:131–146, 1987.

[21] T. Janhunen. Classifying semi-normal default logic on the basis of its expressive power. In M. Gelfond, N. Leone, and G. Pfeifer, editors, *Proceedings of the 5th International Conference on Logic Programming and Non-Monotonic Reasoning, LPNMR'99*, pages 19–33, El Paso, Texas, December 1999. Springer-Verlag. LNAI 1730.

[22] T. Janhunen. On the intertranslatability of non-monotonic logics. *Annals of Mathematics in Artificial Intelligence*, 27(1-4):79–128, 1999.

[23] T. Janhunen. Capturing stationary and regular extensions with Reiter's extensions. In M. Ojeda-Aciego et al., editors, *Logics in Artificial Intelligence, European Workshop, JELIA 2000*, pages 102–117, Málaga, Spain, September/October 2000. Springer-Verlag. LNAI 1919.

[24] T. Janhunen. Comparing the expressive powers of some syntactically restricted classes of logic programs. In J. Lloyd et al., editors, *Computational Logic, First International Conference*, pages 852–866, London, UK, July 2000. Springer-Verlag. LNAI 1861.

[25] T. Janhunen. On the effect of default negation on the expressiveness of disjunctive rules. In T. Eiter, W. Faber, and M. Truszczyński, editors, *Logic Programming and Nonmonotonic Reasoning, Proceedings of the 6th International Conference*, pages 93–106, Vienna, Austria, September 2001. Springer-Verlag. LNAI 2173.

[26] T. Janhunen. Evaluating the effect of semi-normality on the expressiveness of defaults. *Artificial Intelligence*, 144(1–2):233–250, March 2003.

[27] T. Janhunen, I. Niemelä, P. Simons, and J.-H. You. Unfolding partiality and disjunctions in stable model semantics. In A. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 7th International Conference*, pages 411–419, Breckenridge, Colorado, April 2000. Morgan Kaufmann.

[28] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, Oregon, July 1996.

[29] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.

[30] V. Lifschitz, L.R. Tang, and H. Turner. Nested expressions in logic programs. *Annals of Mathematics in Artificial Intelligence*, 25:369–389, 1999.

[31] Fangzhen Lin and Yuting Zhao. ASSAT: Computing answer sets of a logic program by sat solvers. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 112–117, Edmonton, Alberta, Canada, July–August 2002. AAAI.

[32] J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, 1987.

[33] V.W. Marek and V.S. Subrahmanian. The relationship between stable, supported, default and autoepistemic semantics for general logic programs. *Theoretical Computer Science*, 103:365–386, 1992.

[34] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 38:588–619, 1991.

[35] W. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer-Verlag, 1999.

[36] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3,4):241–273, 1999.

[37] D. Nute. Defeasible logic. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, chapter 7, pages 353–395. Oxford Science Publications, 1994.

[38] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978.

[39] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.

[40] J. Schlipf. The expressive powers of the logic programming semantics. *Journal of Computer and System Sciences*, 51:64–86, 1995.

[41] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1–2):181–234, 2002.

[42] V.S. Subrahmanian, D. Nau, and C. Vago. WFS + branch and bound = stable models. *IEEE Transactions on Knowledge and Data Engineering*, 7(3):362–377, 1995.

[43] H. Turner. Strong equivalence made easy: Nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4–5):609–622, 2003.

[44] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23:733–742, 1976.

[45] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, July 1991.

9 APPENDIX: PROOFS

9.1 Correctness of SEL_j

Lemma 9.1 *Let j be the number of bits. If N is a set of atoms such that $\mathsf{c} \notin N$ and $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$ holds for some number n satisfying $0 \leq n < 2^j$, then $\mathrm{LM}(\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$.*

If N is a set of atoms such that $\mathsf{c} \in N$, then $\mathrm{LM}(\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N) = \emptyset$.

PROOF. Suppose that N satisfies the if-part of the first claim. Then consider any atom a and i such that $0 < i \leq j$. Now $\mathsf{a}_i \leftarrow$ belongs to $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N \iff \overline{\mathsf{a}_i} \notin N$ by the structure of $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c}) \iff \mathsf{a}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$ by the requirements imposed on N and (6.3). It is similarly established that $\overline{\mathsf{a}_i} \leftarrow$ is included in $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N \iff \overline{\mathsf{a}_i} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$. Since $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N$ is both positive and atomic, it follows by Lemma 4.9 that $\mathrm{LM}(\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$.

Then assume that $\mathsf{c} \in N$. Since $\sim\mathsf{c}$ appears in each rule of the program $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})$, we obtain $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N = \emptyset$ and $\mathrm{LM}(\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N) = \emptyset$. □

Lemma 9.2 *Let j be the number of bits and N a set of atoms such that $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{LM}(\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N)$.*

If $\mathsf{c} \notin N$, then $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$ holds for $n = \mathrm{val}_j(\mathsf{a}, N)$.

If $\mathsf{c} \in N$, then $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \emptyset$.

PROOF. Suppose that $\mathsf{c} \notin N$ and define $n = \mathrm{val}_j(\mathsf{a}, N)$. Then it holds for any i in the range $0 < i \leq j$ that $\mathsf{a}_i \in N \iff n[i] = 1$ by (6.6) $\iff$ $\mathsf{a}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$ by (6.3). On the other hand, it holds that $\overline{\mathsf{a}_i} \in N \iff \overline{\mathsf{a}_i} \in \mathrm{LM}(\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N) \iff \overline{\mathsf{a}_i} \leftarrow$ belongs to $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N \iff \mathsf{a}_i \notin N$ by the structure of $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c}) \iff n[i] = 0$ by (6.6) $\iff \overline{\mathsf{a}_i} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$ by (6.3). Since i was arbitrary and $\mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a})$ does not contain other atoms than a_i and $\overline{\mathsf{a}_i}$ for each $i \in \{1, \ldots, n\}$, the claim follows.

The case in which $\mathsf{c} \in N$ follows. Since each rule of $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})$ is conditioned by $\sim\mathsf{c}$, we have $\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N = \emptyset$ and $\mathrm{LM}(\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})^N) = \emptyset$. □

9.2 Correctness of NXT_j

It is worth pointing out that the translation $\mathrm{NXT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})$ is based on the following method to increase the value of a binary counter by one. The least significant bit of the counter is always changed. If the bit changed from 1 to 0, then the next significant bit is changed, too, and so on.

Lemma 9.3 *Let j be the number of bits and N a set of atoms such that $\mathsf{c} \notin N$, $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$ for some number n such that $0 \leq n < 2^j$, and $m = n + 1 \bmod 2^j$. Then*

$$\mathsf{b}_i \leftarrow \textit{ belongs to } \mathrm{NXT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})^N \iff \mathsf{b}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b}, m), \textit{ and} \tag{9.1}$$

$$\overline{\mathsf{b}_i} \leftarrow \textit{ belongs to } \mathrm{NXT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})^N \iff \overline{\mathsf{b}_i} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b}, m) \tag{9.2}$$

hold for $i = j$ and for any $0 < i < j$ satisfying

$$\mathsf{b}_{i+1} \in N \iff \mathsf{b}_{i+1} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b}, m) \textit{ and} \tag{9.3}$$

$$\overline{\mathsf{b}_{i+1}} \in N \iff \overline{\mathsf{b}_{i+1}} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b}, m). \tag{9.4}$$

PROOF. Given that $\mathsf{c} \notin N$, the rule $\mathsf{b}_j \leftarrow$ belongs to $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N \iff \mathsf{a}_j \notin N$ by the structure of $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c}) \iff \mathsf{a}_j \notin \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ by the requirements on $N \iff \overline{\mathsf{a}_j} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ by (6.3) $\iff n[j] = 0$ by (6.3) $\iff m[j] = 1$, as $m = n+1 \bmod 2^j$, $\iff \mathsf{b}_j \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ by (6.3). Moreover, it follows by the symmetry present in the sets $\mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ and $\mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ as well as the program $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$ that the rule $\overline{\mathsf{b}_j} \leftarrow$ belongs to $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N \iff \overline{\mathsf{b}_j} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$.

Thus we have established (9.1) and (9.2) when $i = j$. Let us then consider any i such that $0 < i < j$ and the equations (9.3) and (9.4) are satisfied. It follows by the structure of $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$ that $\mathsf{b}_i \leftarrow$ belongs to $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N$

$\iff$ $\mathsf{a}_i \notin N$, $\overline{\mathsf{a}_{i+1}} \notin N$, and $\mathsf{b}_{i+1} \notin N$, **or**
$\overline{\mathsf{a}_i} \notin N$ and $\mathsf{a}_{i+1} \notin N$, **or**
$\overline{\mathsf{a}_i} \notin N$ and $\overline{\mathsf{b}_{i+1}} \notin N$

$\iff$ $\overline{\mathsf{a}_i} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$, $\mathsf{a}_{i+1} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$, and $\overline{\mathsf{b}_{i+1}} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$, **or**
$\mathsf{a}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ and $\overline{\mathsf{a}_{i+1}} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$, **or**
$\mathsf{a}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ and $\mathsf{b}_{i+1} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$
by the requirements on N, (6.3), as well as (9.3) and (9.4)

$\iff$ $n[i \ldots i+1] = 01$ and $m[i+1] = 0$, **or**
$n[i \ldots i+1] = 10$, **or**
$n[i] = 1$ and $m[i+1] = 1$ by (6.3)

$\iff$ $m[i] = 1$, as $m = n+1 \bmod 2^j$,

$\iff \mathsf{b}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$. In this way, we have established (9.1). Due to symmetry, the proof for (9.2) is obtained by systematically exchanging

1. a_i and $\overline{\mathsf{a}_i}$;
2. b_i and $\overline{\mathsf{b}_i}$;
3. $n[i] = 0$ and $n[i] = 1$; as well as
4. $m[i] = 0$ and $m[i] = 1$

in the proof of (9.1). □

Lemma 9.4 *Let j be the number of bits.*

If N is a set of atoms such that $\mathsf{c} \notin N$, $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ for a number n such that $0 \leq n < 2^j$, and $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ for $m = n+1 \bmod 2^j$, then $\mathrm{LM}(\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$.

If N is a set of atoms such that $\mathsf{c} \in N$, then $\mathrm{LM}(\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \emptyset$.

PROOF. Let us assume that N is a set of atoms satisfying the if-part of the first claim. Then the assumptions of Lemma 9.3 are satisfied, and both (9.3) and (9.4) hold for each $0 < i < j$. It follows by Lemma 9.3 that (9.1) and (9.2) hold for any $0 < i \leq j$. Since the reduct $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N$ is both positive and atomic, we conclude $\mathrm{LM}(\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ by Lemma 4.9.

On the other hand, all the rules of $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$ are guarded by the negative literal $\sim\mathsf{c}$. Thus, if $\mathsf{c} \in N$, we have $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N = \emptyset$ which implies $\mathrm{LM}(\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \emptyset$ by Lemma 4.9. □

Lemma 9.5 *Let j be the number of bits and N a set of atoms such that $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{b}) = \mathrm{LM}(\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N)$.*

If $\mathsf{c} \notin N$ and $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ holds for $n = \mathrm{val}_j(\mathsf{a},N)$, then $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{b}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ for $m = n+1 \bmod 2^j$.

If $\mathsf{c} \in N$, then $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{b}) = \emptyset$.

PROOF. Suppose that N satisfies the if-part of the first claim. In the sequel, it is proved by induction on $j - i \geq 0$ that $\mathsf{b}_i \in N \iff \mathsf{b}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$, and that $\overline{\mathsf{b}_i} \in N \iff \overline{\mathsf{b}_i} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$.

Base case: $j - i = 0$ which implies $i = j$. It follows by Lemmas 9.3 and 4.9 and the requirements on N that $\mathsf{b}_j \in N \iff \mathsf{b}_j \in \mathrm{LM}(\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) \iff \mathsf{b}_j \leftarrow$ belongs to $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N \iff \mathsf{b}_j \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$. The fact that $\overline{\mathsf{b}_j} \in N \iff \overline{\mathsf{b}_j} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ follows by symmetry. **Induction step**: $j - i > 0$ which implies $0 < i < j$. Now (9.3) and (9.4) are satisfied by the inductive hypothesis. Thus $\mathsf{b}_i \in N \iff \mathsf{b}_i \in \mathrm{LM}(\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N)$ by the assumptions on $N \iff \mathsf{b}_i \leftarrow$ belongs to $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N$ by Lemma 4.9 $\iff \mathsf{b}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ by Lemma 9.3 and the inductive hypothesis. The fact that $\overline{\mathsf{b}_i} \in N \iff \overline{\mathsf{b}_i} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ is established by symmetry.

If N is a set of atoms such that $\mathsf{c} \in N$, then $\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N = \emptyset$ so that $\mathrm{LM}(\mathrm{NXT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \emptyset$ follows by Lemma 4.9. □

9.3 Correctness of FIX_j

Lemma 9.6 *Let j be the number of bits and $0 \leq n < 2^j$. If N is a set of atoms such that $\mathsf{c} \notin N$, then $\mathrm{LM}(\mathrm{FIX}_j(\mathsf{a},n,\mathsf{c})^N) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$.*

If N is a set of atoms such that $\mathsf{c} \in N$, then $\mathrm{LM}(\mathrm{FIX}_j(\mathsf{a},n,\mathsf{c})^N) = \emptyset$.

PROOF. Let N be a set of atoms such that $\mathsf{c} \notin N$ and $0 < i \leq j$. The structure of $\mathrm{FIX}_j(\mathsf{a},n,\mathsf{c})$ and Lemma 4.9 imply that $\overline{\mathsf{a}_i} \in \mathrm{LM}(\mathrm{FIX}_j(\mathsf{a},n,\mathsf{c})^N)$ (resp. $\mathsf{a}_i \in \mathrm{LM}(\mathrm{FIX}_j(\mathsf{a},n,\mathsf{c})^N)$) $\iff$ the rule $\overline{\mathsf{a}_i} \leftarrow$ (resp. $\mathsf{a}_i \leftarrow$) belongs to $\mathrm{FIX}_j(\mathsf{a},n,\mathsf{c})^N \iff n[i] = 0$ (resp. $n[i] = 1$) $\iff \overline{\mathsf{a}_i} \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ (resp. $\mathsf{a}_i \in \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$). Hence the claim. In case that $\mathsf{c} \in N$, we have $\mathrm{FIX}_j(\mathsf{a},n,\mathsf{c})^N = \emptyset$ so that the claim follows by Lemma 4.9. □

9.4 Correctness of LT_j

Lemma 9.7 *Let j be the number of bits and N a set of atoms such that $\mathsf{c} \notin N$, $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a},n)$ for $0 \leq n < 2^j$, and $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{b}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b},m)$ for $0 \leq m < 2^j$.*

If $0 < i \leq j$ and in addition, $i < j$ implies

$$\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_{i+1}} \in N \iff \overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_{i+1}} \in \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a},n,\mathsf{b},m), \tag{9.5}$$

then

$$\mathsf{lt}(\mathsf{a},\mathsf{b})_i \leftarrow \text{ belongs to } \mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N \iff \mathsf{lt}(\mathsf{a},\mathsf{b})_i \in \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a},n,\mathsf{b},m). \tag{9.6}$$

If $0 < i \leq j$ and

$$\mathsf{lt}(\mathsf{a},\mathsf{b})_i \in N \iff \mathsf{lt}(\mathsf{a},\mathsf{b})_i \in \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a},n,\mathsf{b},m), \tag{9.7}$$

then

$$(9.8)\quad \overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_i} \leftarrow \textit{ belongs to } \mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N \iff \overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_i} \in \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m).$$

PROOF. If $i = j$, then the rule $\mathsf{lt}(\mathsf{a},\mathsf{b})_j \leftarrow$ is included in $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N \iff \mathsf{a}_j \notin N$ and $\overline{\mathsf{b}_j} \notin N \iff \overline{\mathsf{a}_j} \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n)$ and $\mathsf{b}_j \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m)$ by the requirements on $N \iff n[j] = 0$ and $m[j] = 1$ by (6.3) $\iff n[j] < m[j]$ $\iff \mathsf{lt}(\mathsf{a},\mathsf{b})_j \in \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m)$ by (6.4). Thus (9.6) holds when $i = j$.

If $0 < i < j$, then (9.5) holds and $\mathsf{lt}(\mathsf{a},\mathsf{b})_i \leftarrow$ belongs to $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N$

$\iff$ $\mathsf{a}_i \notin N$ and $\overline{\mathsf{b}_i} \notin N$, **or**
$\mathsf{a}_i \notin N$ and $\mathsf{b}_i \notin N$, and $\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_{i+1}} \notin N$, **or**
$\overline{\mathsf{a}_i} \notin N$ and $\overline{\mathsf{b}_i} \notin N$, and $\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_{i+1}} \notin N$
by the structure of $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$

$\iff$ $\overline{\mathsf{a}_i} \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n)$ and $\mathsf{b}_i \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m)$, **or**
$\overline{\mathsf{a}_i} \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n)$, $\overline{\mathsf{b}_i} \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m)$, and
$\mathsf{lt}(\mathsf{a},\mathsf{b})_{i+1} \in \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m)$, **or**
$\mathsf{a}_i \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n)$, $\mathsf{b}_i \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m)$, and
$\mathsf{lt}(\mathsf{a},\mathsf{b})_{i+1} \in \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m)$
by the requirements on N, (9.5), as well as (6.4)

$\iff$ $n[i] = 0$ and $m[i] = 1$, **or**
$n[i] = m[i] = 0$ and $n[i+1 \ldots j] < m[i+1 \ldots j]$, **or**
$n[i] = m[i] = 1$ and $n[i+1 \ldots j] < m[i+1 \ldots j]$ by (6.4)

$\iff$ $n[i \ldots j] < m[i \ldots j]$

$\iff$ $\mathsf{lt}(\mathsf{a},\mathsf{b})_i \in \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m)$.

Thus we have established (9.6). Then consider any $0 < i \leq j$ and assume that (9.7) holds. Now the rule $\overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_i} \leftarrow$ belongs to $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N \iff \mathsf{lt}(\mathsf{a},\mathsf{b})_i \notin N$ by the structure of $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c}) \iff \mathsf{lt}(\mathsf{a},\mathsf{b})_i \notin \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m)$ by (9.7) $\iff \overline{\mathsf{lt}(\mathsf{a},\mathsf{b})_i} \in \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m)$ by (6.4). Thus (9.8) holds. □

Lemma 9.8 *Let j be the number of bits.*

If N is a set of atoms such that $\mathsf{c} \notin N$, $N \cap \mathrm{Hb}^{\mathrm{ctr}}_j(\mathsf{a}) = \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n)$ for $0 \leq n < 2^j$, $N \cap \mathrm{Hb}^{\mathrm{ctr}}_j(\mathsf{b}) = \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m)$ for $0 \leq m < 2^j$, and $N \cap \mathrm{head}(\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})) = \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m)$, then

$$\mathrm{LM}(\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \mathrm{AT}^{\mathrm{lt}}_j(\mathsf{a},n,\mathsf{b},m).$$

If N is a set of atoms such that $\mathsf{c} \in N$, then $\mathrm{LM}(\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \emptyset$.

PROOF. Suppose that $\mathsf{c} \notin N$. Consequently, it is easy to see that (9.5) is satisfied by every $0 < i < j$ and (9.7) is satisfied by every $0 < i \leq j$. Then (9.6) and (9.8) hold for all $0 < i \leq j$. Since $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N$ is atomic, it follows by Lemma 4.9 that $\mathrm{LM}(\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n,\mathsf{b},m)$. On the other hand, if $\mathsf{c} \in N$, then $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N = \emptyset$ by the structure of $\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})$ so that $\mathrm{LM}(\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \emptyset$. □

Lemma 9.9 *Let j be the number of bits and N a set of atoms such that $N \cap \mathrm{head}(\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})) = \mathrm{LM}(\mathrm{LT}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N)$.*

If $\mathsf{c} \notin N$, $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$ *where* $n = \mathrm{val}_j(\mathsf{a}, N)$, *and* $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{b}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b}, m)$ *where* $m = \mathrm{val}_j(\mathsf{b}, N)$, *then*

$$N \cap \mathrm{head}(\mathrm{LT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})) = \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a}, n, \mathsf{b}, m).$$

If $\mathsf{c} \in N$, *then* $N \cap \mathrm{head}(\mathrm{LT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})) = \emptyset$.

PROOF. Suppose that N satisfies the if-part of the first claim. We use induction on $j - i \geq 0$ to establish that $\mathsf{lt}(\mathsf{a}, \mathsf{b})_i \in N \iff \mathsf{lt}(\mathsf{a}, \mathsf{b})_i \in \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a}, n, \mathsf{b}, m)$, and $\overline{\mathsf{lt}(\mathsf{a}, \mathsf{b})}_i \in N \iff \overline{\mathsf{lt}(\mathsf{a}, \mathsf{b})}_i \in \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a}, n, \mathsf{b}, m)$.

Base case: $j - i = 0$ which implies $i = j$. The structure of $\mathrm{LT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})$ implies together with Lemmas 9.7 and 4.9 that $\mathsf{lt}(\mathsf{a}, \mathsf{b})_j \in N \iff \mathsf{lt}(\mathsf{a}, \mathsf{b})_j \in \mathrm{LM}(\mathrm{LT}_j(\mathsf{a}, \mathsf{b})^N) \iff \mathsf{lt}(\mathsf{a}, \mathsf{b})_j \leftarrow$ belongs to $\mathrm{LT}_j(\mathsf{a}, \mathsf{b})^N \iff \mathsf{lt}(\mathsf{a}, \mathsf{b})_j \in \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a}, n, \mathsf{b}, m)$. Thus (9.7) holds for $i = j$ so that the other equivalence is similarly obtained using (9.8). **Induction step**: $j - i > 0$ which implies $0 < i < j$. Note that (9.5) is satisfied by the inductive hypothesis. Thus (9.6) holds by Lemma 9.7. It follows by Lemma 4.9 that $\mathsf{lt}(\mathsf{a}, \mathsf{b})_i \in N \iff \mathsf{lt}(\mathsf{a}, \mathsf{b})_i \in \mathrm{LM}(\mathrm{LT}_j(\mathsf{a}, \mathsf{b})^N) \iff \mathsf{lt}(\mathsf{a}, \mathsf{b})_i \leftarrow$ belongs to $\mathrm{LT}_j(\mathsf{a}, \mathsf{b})^N \iff \mathsf{lt}(\mathsf{a}, \mathsf{b})_i \in \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a}, n, \mathsf{b}, m)$. Thus we have established (9.7) which implies (9.8) by Lemma 9.7. Then $\overline{\mathsf{lt}(\mathsf{a}, \mathsf{b})_i} \in N \iff \overline{\mathsf{lt}(\mathsf{a}, \mathsf{b})_i} \in \mathrm{AT}_j^{\mathrm{lt}}(\mathsf{a}, n, \mathsf{b}, m)$ can be established similarly to (9.7). □

9.5 Correctness of EQ_j

Lemma 9.10 *Let* j *be the number of bits and* N *a set of atoms such that* $\mathsf{c} \notin N$, $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{a}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{a}, n)$ *for* $0 \leq n < 2^j$, *and* $N \cap \mathrm{Hb}_j^{\mathrm{ctr}}(\mathsf{b}) = \mathrm{AT}_j^{\mathrm{ctr}}(\mathsf{b}, m)$ *for* $0 \leq m < 2^j$. *If*

$$\overline{\mathsf{eq}(\mathsf{a}, \mathsf{b})} \in N \iff \overline{\mathsf{eq}(\mathsf{a}, \mathsf{b})} \in \mathrm{AT}_j^{\mathrm{eq}}(\mathsf{a}, n, \mathsf{b}, m), \tag{9.9}$$

then

$$\mathsf{eq}(\mathsf{a}, \mathsf{b}) \leftarrow \textit{ belongs to } \mathrm{EQ}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})^N \iff \mathsf{eq}(\mathsf{a}, \mathsf{b}) \in \mathrm{AT}_j^{\mathrm{eq}}(\mathsf{a}, n, \mathsf{b}, m). \tag{9.10}$$

Moreover,

$$\overline{\mathsf{eq}(\mathsf{a}, \mathsf{b})} \leftarrow \textit{ belongs to } \mathrm{EQ}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})^N \iff \overline{\mathsf{eq}(\mathsf{a}, \mathsf{b})} \in \mathrm{AT}_j^{\mathrm{eq}}(\mathsf{a}, n, \mathsf{b}, m). \tag{9.11}$$

PROOF. Suppose that (9.9) holds. Now the rule $\mathsf{eq}(\mathsf{a}, \mathsf{b}) \leftarrow$ belongs to $\mathrm{EQ}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})^N \iff \overline{\mathsf{eq}(\mathsf{a}, \mathsf{b})} \notin N$ by the structure of $\mathrm{EQ}_j(\mathsf{a}, \mathsf{b}, \mathsf{c}) \iff \overline{\mathsf{eq}(\mathsf{a}, \mathsf{b})} \notin \mathrm{AT}_j^{\mathrm{eq}}(\mathsf{a}, n, \mathsf{b}, m)$ by (9.9) $\iff \mathsf{eq}(\mathsf{a}, \mathsf{b}) \in \mathrm{AT}_j^{\mathrm{eq}}(\mathsf{a}, n, \mathsf{b}, m)$ by (6.5). On the other hand, the rule $\overline{\mathsf{eq}(\mathsf{a}, \mathsf{b})} \leftarrow$ is included in $\mathrm{EQ}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})^N$

$$
\begin{array}{ll}
\iff & \exists i \in \{1,\ldots,j\} \text{ such that } \mathsf{a}_i \notin N \text{ and } \overline{\mathsf{b}_i} \notin N, \textbf{ or} \\
 & \qquad\qquad\qquad\qquad \overline{\mathsf{a}_i} \notin N \text{ and } \mathsf{b}_i \notin N \\
\iff & \exists i \in \{1,\ldots,j\} \text{ such that } \overline{\mathsf{a}_i} \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n) \text{ and } \mathsf{b}_i \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m), \textbf{ or} \\
 & \qquad\qquad\qquad\qquad \mathsf{a}_i \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n) \text{ and } \overline{\mathsf{b}_i} \in \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m) \\
\iff & \exists i \in \{1,\ldots,j\} \text{ such that } n[i]=0 \text{ and } m[i]=1, \textbf{ or} \\
 & \qquad\qquad\qquad\qquad n[i]=1 \text{ and } m[i]=0 \\
\iff & n[1\ldots j] \neq m[1\ldots j] \\
\iff & n \neq m \\
\iff & \overline{\mathsf{eq}(\mathsf{a},\mathsf{b})} \in \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},m). \qquad \square
\end{array}
$$

Lemma 9.11 *Let j be the number of bits.*

If N is a set of atoms such that $\mathsf{c} \notin N$, $N \cap \mathrm{Hb}^{\mathrm{ctr}}_j(\mathsf{a}) = \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n)$ for $0 \leq n < 2^j$, $N \cap \mathrm{Hb}^{\mathrm{ctr}}_j(\mathsf{b}) = \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m)$ for $0 \leq m < 2^j$, and $N \cap \mathrm{head}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})) = \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},m)$, then

$$\mathrm{LM}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},n).$$

If N is a set of atoms such that $\mathsf{c} \in N$, then $\mathrm{LM}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \emptyset$.

PROOF. Suppose that N satisfies the if-part of the first claim. Consequently, the prerequisites of Lemma 9.10 are satisfied — including the equation (9.9). Thus (9.10) and (9.11) follow by Lemma 9.10. Since $\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N$ is atomic, we obtain $\mathrm{LM}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},m)$ by Lemma 4.9. On the other hand, if $\mathsf{c} \in N$, then $\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N = \emptyset$, as $\sim\mathsf{c}$ is included in the body of each rule in $\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})$. It follows that $\mathrm{LM}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \emptyset$. $\square$

Lemma 9.12 *Let j be the number of bits and N a set of atoms such that $N \cap \mathrm{head}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})) = \mathrm{LM}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N)$.*

If $\mathsf{c} \notin N$, $N \cap \mathrm{Hb}^{\mathrm{ctr}}_j(\mathsf{a}) = \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{a},n)$ where $n = \mathrm{val}_j(\mathsf{a},N)$, and $N \cap \mathrm{Hb}^{\mathrm{ctr}}_j(\mathsf{b}) = \mathrm{AT}^{\mathrm{ctr}}_j(\mathsf{b},m)$ where $m = \mathrm{val}_j(\mathsf{b},N)$, then

$$N \cap \mathrm{head}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})) = \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},m).$$

If $\mathsf{c} \in N$, then $N \cap \mathrm{head}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})) = \emptyset$.

PROOF. Suppose that N satisfies the if-part of the first claim so that the prerequisites of Lemma 9.10 are met. Then $\overline{\mathsf{eq}(\mathsf{a},\mathsf{b})} \in N \iff \overline{\mathsf{eq}(\mathsf{a},\mathsf{b})} \in \mathrm{LM}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b})^N) \iff \overline{\mathsf{eq}(\mathsf{a},\mathsf{b})} \leftarrow$ belongs to $\mathrm{EQ}_j(\mathsf{a},\mathsf{b})^N \iff \overline{\mathsf{eq}(\mathsf{a},\mathsf{b})} \in \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},m)$ by Lemma 4.9 and (9.11) in Lemma 9.10. This is how (9.9) is established first, and then the fact that $\mathsf{eq}(\mathsf{a},\mathsf{b}) \in N \iff \mathsf{eq}(\mathsf{a},\mathsf{b}) \in \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},m)$ follows similarly using (9.10). Thus $\mathrm{LM}(\mathrm{EQ}_j(\mathsf{a},\mathsf{b},\mathsf{c})^N) = \mathrm{AT}^{\mathrm{eq}}_j(\mathsf{a},n,\mathsf{b},m)$.

The latter claim follows by the same arguments as in Lemma 9.11. $\square$

9.6 Correctness of $\mathrm{Tr}_{\mathrm{SUPP}}(P)$

Lemma 9.13 *Let P be a normal program, $M \subseteq \mathrm{Hb}(P)$ an interpretation of P, and $\#$ a function from $M \cup \mathrm{SR}(P,M)$ to $\{0,\ldots,2^{\nabla P}-1\}$, and $N = \mathrm{Ext}_{\mathrm{AT}}(P,M,\#)$. The respective intersections of N with $\mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$, $\mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$, $\mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$, and $\mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$ are the sets*

$\mathrm{Ext}_{\mathrm{SUPP}}(P, M)$, $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$, $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$, *and* $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$.

PROOF. Definitions 6.14 and 6.8 imply that $\mathrm{Ext}_{\mathrm{SUPP}}(P, M)$ is contained in $\mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. The same can be stated about the other three extension operators $\mathrm{Ext}_{\mathrm{CTR}}$, $\mathrm{Ext}_{\mathrm{MAX}}$, and $\mathrm{Ext}_{\mathrm{MIN}}$, and the respective translations of P by Definitions 6.15, 6.9, 6.10, and 6.11. In addition to this, the four sets of head atoms are disjoint by Definitions 6.8, 6.9, 6.10, and 6.11. The rest follows by the definition of $\mathrm{Ext}_{\mathrm{AT}}$ in Definition 6.16. □

Proposition 9.14 *Let P be a normal program. If M is a supported model of P, $\#$ a level numbering w.r.t. M, and $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$, then $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P)) = \mathrm{Ext}_{\mathrm{SUPP}}(P, M) = \mathrm{LM}(\mathrm{Tr}_{\mathrm{SUPP}}(P)^N)$.*

PROOF. Let M be a supported model of P and $\#$ a level numbering w.r.t. M. Recall the translation $\mathrm{Tr}_{\mathrm{SUPP}}(P)$ from Definition 6.8 and the set of atoms $N = \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$ from Definition 6.14. The first equality $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P)) = \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$ follows directly by Lemma 9.13. The second equality is established by showing that a rule $r \in \mathrm{Tr}_{\mathrm{SUPP}}(P)^N \iff \mathrm{head}(r) \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$. There are four types of rules in the translation $\mathrm{Tr}_{\mathrm{SUPP}}(P)$ that have to be checked in this respect.

1. Consider any atom $\mathsf{a} \in \mathrm{Hb}(P)$ for which $\mathrm{Tr}_{\mathrm{SUPP}}(P)$ contains the rule $\overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a}$. Now $\overline{\mathsf{a}} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N \iff \mathsf{a} \notin N \iff \mathsf{a} \notin M$, as $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P)) = \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$, $\iff \overline{\mathsf{a}} \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$.

2. Then let $r \in P$ for which $\mathsf{bt}(r) \leftarrow {\sim}\overline{\mathrm{body}^+(r)}, {\sim}\mathrm{body}^-(r)$ is included in the translation $\mathrm{Tr}_{\mathrm{SUPP}}(P)$. Consequently, the rule $\mathsf{bt}(r) \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N \iff N \models {\sim}\overline{\mathrm{body}^+(r)} \cup {\sim}\mathrm{body}^-(r) \iff \overline{\mathrm{body}^+(r)} \cap N = \emptyset$ and $\mathrm{body}^-(r) \cap N = \emptyset \iff \mathrm{body}^+(r) \subseteq M$ and $\mathrm{body}^-(r) \cap M = \emptyset$, since $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P)) = \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$ and $\mathrm{body}^+(r) \cup \mathrm{body}^-(r) \subseteq \mathrm{Hb}(P)$, $\iff M \models \mathrm{body}(r) \iff r \in \mathrm{SR}(P, M) \iff \mathsf{bt}(r) \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$.

3. Then consider the rule $\overline{\mathsf{bt}(r)} \leftarrow {\sim}\mathsf{bt}(r)$ associated with a rule $r \in P$ and included in $\mathrm{Tr}_{\mathrm{SUPP}}(P)$. Now $\overline{\mathsf{bt}(r)} \leftarrow$ is included in $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N \iff \mathsf{bt}(r) \notin N \iff r \notin \mathrm{SR}(P, M)$, as $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P)) = \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$, $\iff \overline{\mathsf{bt}(r)} \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$.

4. Finally, there is a rule $\mathrm{head}(r) \leftarrow {\sim}\overline{\mathsf{bt}(r)} \in \mathrm{Tr}_{\mathrm{SUPP}}(P)$ for each $r \in P$. Then for any $\mathsf{a} \in \mathrm{head}(P)$, the rule $\mathsf{a} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N \iff \exists r \in P$ such that $\mathrm{head}(r) = \mathsf{a}$ and $\overline{\mathsf{bt}(r)} \notin N \iff \exists r \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r) = \mathsf{a}$, as $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ equals to $\mathrm{Ext}_{\mathrm{SUPP}}(P, M)$, $\iff \mathsf{a} \in M$, since M is a supported model of P, $\iff \mathsf{a} \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$.

It follows by Lemma 4.9 that $\mathrm{LM}(\mathrm{Tr}_{\mathrm{SUPP}}(P)^N) = \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$. □

Proposition 9.15 *Let P be a normal program. If N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, then $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P)) = \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$ for $M = N \cap \mathrm{Hb}(P)$.*

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, i.e. $N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(P)^N)$. Recall the translation $\mathrm{Tr}_{\mathrm{SUPP}}(P)$ and $\mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ from Definition 6.8. In the sequel, we establish for all atoms $\mathsf{a} \in \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ that $\mathsf{a} \in N \iff \mathsf{a} \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$. Four cases have to be analyzed.

1. Recall that $\mathrm{head}(P) \subseteq \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. The definition of M implies for all $\mathsf{a} \in \mathrm{head}(P)$ that $\mathsf{a} \in N \iff \mathsf{a} \in M \iff \mathsf{a} \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$.

2. Then consider any $\mathsf{a} \in \mathrm{Hb}(P)$ for which $\overline{\mathsf{a}} \in \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. It follows that $\overline{\mathsf{a}} \in N \iff \overline{\mathsf{a}} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(P)^N) \iff \overline{\mathsf{a}} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{AT}}(P)^N$ by Lemma 4.9 $\iff \overline{\mathsf{a}} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N$ by the structure of $\mathrm{Tr}_{\mathrm{AT}}(P) \iff \mathsf{a} \notin N$ by the structure of $\mathrm{Tr}_{\mathrm{SUPP}}(P) \iff \mathsf{a} \notin M$ by the definition of M, as $\mathsf{a} \in \mathrm{Hb}(P)$, $\iff \overline{\mathsf{a}} \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$.

3. Let us then analyze any $r \in P$ and the respective atom $\mathsf{bt}(r)$ included in $\mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. Recall that $\mathsf{bt}(r) \leftarrow \sim\overline{\mathrm{body}^+(r)}, \sim\mathrm{body}^-(r)$ belongs to $\mathrm{Tr}_{\mathrm{SUPP}}(P)$. The relationship established in the previous item implies for any $\mathsf{a} \in \mathrm{body}^+(r)$ that (i) $\overline{\mathsf{a}} \notin N \iff \mathsf{a} \in M$. On the other hand, the definition of M implies that (ii) $\mathsf{a} \notin N \iff \mathsf{a} \notin M$ for any $\mathsf{a} \in \mathrm{body}^-(r)$. Thus $\mathsf{bt}(r) \in N \iff \mathsf{bt}(r) \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(P)^N) \iff \mathsf{bt}(r) \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{AT}}(P)^N$ by Lemma 4.9 $\iff \mathsf{bt}(r) \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N$ by the structure of $\mathrm{Tr}_{\mathrm{AT}}(P) \iff N \models \sim\mathrm{body}^+(r)$ and $N \models \sim\mathrm{body}^-(r) \iff M \models \mathrm{body}(r)$ by (i) and (ii) above $\iff r \in \mathrm{SR}(P, M) \iff \mathsf{bt}(r) \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$.

4. The case of the complementary atom $\overline{\mathsf{bt}(r)} \in \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ follows. Recall that $r \in P$ and that the rule $\overline{\mathsf{bt}(r)} \leftarrow \sim\mathsf{bt}(r)$ is included in $\mathrm{Tr}_{\mathrm{SUPP}}(P)$. Thus $\overline{\mathsf{bt}(r)} \in N \iff \overline{\mathsf{bt}(r)} \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(P)^N) \iff \overline{\mathsf{bt}(r)} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{AT}}(P)^N$ by Lemma 4.9 $\iff \overline{\mathsf{bt}(r)} \leftarrow$ belongs to $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N$ by the structure of $\mathrm{Tr}_{\mathrm{AT}}(P) \iff \mathsf{bt}(r) \notin N \iff r \notin \mathrm{SR}(P, M)$, as shown above, $\iff \overline{\mathsf{bt}(r)} \in \mathrm{Ext}_{\mathrm{SUPP}}(P, M)$. □

Proposition 9.16 *Let P be a normal program. If N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, then $M = N \cap \mathrm{Hb}(P)$ is a supported model of P.*

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$ and define M as above. Suppose that $M \not\models P$. Then there is a rule $r \in P$ such that $M \models \mathrm{body}(r)$ but $M \not\models \mathrm{head}(r)$. Thus $r \in \mathrm{SR}(P, M)$ and it follows by Proposition 9.15 that $\mathsf{bt}(r) \in N$ and $\overline{\mathsf{bt}(r)} \notin N$. But then $\mathrm{head}(r) \leftarrow$ belongs to the reduct $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N \subseteq \mathrm{Tr}_{\mathrm{AT}}(P)^N$ implying that $\mathrm{head}(r) \in \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(P)^N) = N$. As $\mathrm{head}(r) \in \mathrm{Hb}(P)$, we obtain $\mathrm{head}(r) \in M$. That is, $M \models \mathrm{head}(r)$, a contradiction. Hence $M \models P$ is the case.

Let us then assume that M is not a supported model of P. Then there is an atom $\mathsf{a} \in M$ such that for every rule $r \in P$ with $\mathrm{head}(r) = \mathsf{a}$, we have $M \not\models \mathrm{body}(r)$, i.e. $r \notin \mathrm{SR}(P, M)$. Then let r be any rule with $\mathrm{head}(r) = \mathsf{a}$. Recall that $\mathrm{head}(r) \leftarrow \sim\overline{\mathsf{bt}(r)}$ is included in $\mathrm{Tr}_{\mathrm{SUPP}}(P)$. It follows Proposition 9.15 that $r \notin \mathrm{SR}(P, M) \iff \overline{\mathsf{bt}(r)} \in N \iff \mathrm{head}(r) \leftarrow$ is not included in $\mathrm{Tr}_{\mathrm{SUPP}}(P)^N \iff \mathrm{head}(r) \leftarrow$ does not belong to $\mathrm{Tr}_{\mathrm{AT}}(P)^N$.

To summarize, it follows by Lemma 4.9 that $\mathsf{a} \notin \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(P)^N)$. Thus $\mathsf{a} \notin N$ by the stability of N. Then the definition of M implies $\mathsf{a} \notin M$, a contradiction. □

9.7 Correctness of $\mathrm{Tr}_{\mathrm{CTR}}(P)$

Proposition 9.17 *Let P be a normal program. If M is a stable model of P, $\#$ the corresponding level numbering w.r.t. M, and $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$, then $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P)) = \mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = \mathrm{LM}(\mathrm{Tr}_{\mathrm{CTR}}(P)^N)$.*

PROOF. Let M be a stable model of P, $\#$ the corresponding level numbering, as implied by Theorem 5.8, and $N = \mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$. Now $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P)) = \mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$ holds directly by Lemma 9.13. The translation $\mathrm{Tr}_{\mathrm{CTR}}(P)$ contains four kinds of subprograms as listed in Definition 6.9. We shall establish that N is locally stable w.r.t. each of these.

1. Consider a subprogram $Q_{\mathsf{ctr(a)}} = \mathrm{SEL}_{\nabla P}(\mathsf{ctr(a)}, \overline{\mathsf{a}})$ associated with an atom $\mathsf{a} \in \mathrm{Hb}(P)$. Now $\mathrm{head}(Q_{\mathsf{ctr(a)}}) = \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr(a)})$ is one of the subsets that form $\mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$. Two cases arise depending on the membership of $\overline{\mathsf{a}}$ in N. If $\overline{\mathsf{a}} \notin N$, then $\mathsf{a} \in M$ by the definition of N. The definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ implies that $N \cap \mathrm{head}(Q_{\mathsf{ctr(a)}}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr(a)}, \#\mathsf{a})$. Then $\mathrm{LM}(Q^N_{\mathsf{ctr(a)}}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr(a)}, \#\mathsf{a})$ follows by Lemma 9.1. On the other hand, if $\overline{\mathsf{a}} \in N$, we have $\mathsf{a} \notin M$ by the definition of N. Then $N \cap \mathrm{head}(Q_{\mathsf{ctr(a)}}) = \emptyset$ by the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ and $\mathrm{LM}(Q^N_{\mathsf{ctr(a)}}) = \emptyset$ by Lemma 9.1, as $\overline{\mathsf{a}} \in N$ holds. Thus N is locally stable w.r.t. $Q_{\mathsf{ctr(a)}}$ in both cases.

2. Let us then analyze the program $Q_{\mathsf{nxt(a)}} = \mathrm{NXT}_{\nabla P}(\mathsf{ctr(a)}, \mathsf{nxt(a)}, \overline{\mathsf{a}})$ associated with $\mathsf{a} \in \mathrm{Hb}(P)$. Now $\mathrm{head}(Q_{\mathsf{nxt(a)}}) = \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt(a)})$ is one part of $\mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$. Like above, we have two cases to consider. Suppose that $\overline{\mathsf{a}} \notin N$ is the case. As noted in the previous item, $\mathsf{a} \in M$ and $N \cap \mathrm{head}(Q_{\mathsf{ctr(a)}}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr(a)}, \#\mathsf{a})$. In addition, $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt(a)}, \#\mathsf{a} + 1 \bmod 2^{\nabla P})$ by the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$, since $\mathrm{head}(Q_{\mathsf{nxt(a)}}) = \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt(a)})$. Proposition 6.6 implies that $\#\mathsf{a}+1 \bmod 2^{\nabla P} = \#\mathsf{a}+1$. Then, by Lemma 9.4, it holds that $\mathrm{LM}(Q^N_{\mathsf{nxt(a)}}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt(a)}, \#\mathsf{a} + 1)$. On the other hand, if $\overline{\mathsf{a}} \in N$, then $\mathsf{a} \notin M$ by the definition of N. Then the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ implies $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \emptyset$ and $\mathrm{LM}(Q^N_{\mathsf{nxt(a)}}) = \emptyset$ follows by Lemma 9.4. Thus N is locally stable w.r.t. $Q_{\mathsf{nxt(a)}}$.

3. Our next concern is the program $Q_{\mathsf{ctr}(r)} = \mathrm{SEL}_{\nabla P}(\mathsf{ctr}(r), \overline{\mathsf{bt}(r)})$ associated with a rule $r \in P$ such that $\mathrm{body}^+(r) \neq \emptyset$. Note that $\mathrm{head}(Q_{\mathsf{ctr}(r)}) = \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r))$ is a part of $\mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$. Since $\overline{\mathsf{bt}(r)}$ appears as the control atom of the subprogram, two cases arise depending its membership in N. If $\overline{\mathsf{bt}(r)} \notin N$, then $r \in \mathrm{SR}(P, M)$ by the definition of N. It follows that $N \cap \mathrm{head}(Q_{\mathsf{ctr}(r)}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#r)$ by the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$. Thus, we obtain $\mathrm{LM}(Q^N_{\mathsf{ctr}(r)}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#r)$ by Lemma 9.1. The case

$\overline{\mathsf{bt}(r)} \in N$ is covered as follows. Now $r \notin \mathrm{SR}(P, M)$ by the definition of N. The definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ implies $N \cap \mathrm{head}(Q_{\mathsf{ctr}(r)}) = \emptyset$. This is in harmony with $\mathrm{LM}(Q^N_{\mathsf{ctr}(r)}) = \emptyset$ which holds by Lemma 9.1. To conclude, N is locally stable w.r.t. $Q_{\mathsf{ctr}(r)}$.

4. The last subprogram $Q_{\mathsf{ctr}(r)} = \mathrm{FIX}_{\nabla P}(\mathsf{ctr}(r), 1, \overline{\mathsf{bt}(r)})$ is associated with a rule $r \in P$ such that $\mathrm{body}^+(r) = \emptyset$. Recall that $\mathrm{head}(Q_{\mathsf{ctr}(r)}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), 1)$. There are two possibilities. If $r \in \mathrm{SR}(P, M)$, then $\#r = 1$ by Definition 5.2. The definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ implies $\overline{\mathsf{bt}(r)} \notin N$. Then Lemma 9.6 implies that $\mathrm{LM}(Q^N_{\mathsf{ctr}(r)}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), 1)$. This complies with the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ by which we obtain $N \cap \mathrm{head}(Q_{\mathsf{ctr}(r)}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), 1)$. On the other hand, if $r \notin \mathrm{SR}(P, M)$, then $\overline{\mathsf{bt}(r)} \in N$ by the definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. Then $N \cap \mathrm{head}(Q_{\mathsf{ctr}(r)}) = \emptyset$ by the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ and accordingly $\mathrm{LM}(Q^N_{\mathsf{ctr}(r)}) = \emptyset$ by Lemma 9.6. Thus N is locally stable w.r.t. $Q_{\mathsf{ctr}(r)}$ in both cases.

It follows by Theorem 6.22 that N is locally stable w.r.t. $\mathrm{Tr}_{\mathrm{CTR}}(P)$, i.e. $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P)) = \mathrm{LM}(\mathrm{Tr}_{\mathrm{CTR}}(P)^N)$. □

Proposition 9.18 *Let P be a normal program. If N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, then $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P)) = \mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$ for $M = N \cap \mathrm{Hb}(P)$ and the function $\# : M \cup \mathrm{SR}(P, M) \to \{0, \ldots, 2^{\nabla P} - 1\}$ from Definition 6.18.*

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$ so that $N = \mathrm{LM}(\mathrm{Tr}_{\mathrm{AT}}(P)^N)$ holds. Let $M = N \cap \mathrm{Hb}(P)$ and define the function $\#$ as in Definition 6.18. Proposition 9.15 implies for any $\mathsf{a} \in \mathrm{Hb}(P)$ and $r \in P$ that

$$\mathsf{a} \in M \iff \overline{\mathsf{a}} \notin N, \text{ and} \tag{9.12}$$

$$r \in \mathrm{SR}(P, M) \iff \overline{\mathsf{bt}(r)} \notin N. \tag{9.13}$$

Then recall the translation $\mathrm{Tr}_{\mathrm{CTR}}(P)$ and $\mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ from Definition 6.9. The set of head atoms $\mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ is partitioned as follows.

1. First, we have $\mathrm{head}(Q_{\mathsf{ctr}(\mathsf{a})}) = \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(\mathsf{a}))$ for any $\mathsf{a} \in \mathrm{Hb}(P)$ and the respective subprogram $Q_{\mathsf{ctr}(\mathsf{a})} = \mathrm{SEL}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), \overline{\mathsf{a}})$. Since N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, it follows by Theorem 6.22 that N is locally stable w.r.t. $Q_{\mathsf{ctr}(\mathsf{a})} \subseteq \mathrm{Tr}_{\mathrm{AT}}(P)$, i.e. $N \cap \mathrm{head}(Q_{\mathsf{ctr}(\mathsf{a})}) = \mathrm{LM}(Q^N_{\mathsf{ctr}(\mathsf{a})})$. Two cases arise for our consideration. If $\mathsf{a} \in M$, then $\overline{\mathsf{a}} \notin N$ by (9.12) and $N \cap \mathrm{head}(Q_{\mathsf{ctr}(\mathsf{a})}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), \#\mathsf{a})$ follows by Lemma 9.2, since $\#\mathsf{a} = \mathrm{val}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), N)$ holds by the definition of $\#\mathsf{a}$ for $\mathsf{a} \in M$. On the other hand, if $\mathsf{a} \notin M$, we have $\overline{\mathsf{a}} \in N$ by (9.12). It follows by Lemma 9.2 that $N \cap \mathrm{head}(Q_{\mathsf{ctr}(\mathsf{a})}) = \emptyset$. To conclude, $N \cap \mathrm{head}(Q_{\mathsf{ctr}(\mathsf{a})})$ is compatible with the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$ for each $\mathsf{a} \in \mathrm{Hb}(P)$.

2. Let us then analyze the program $Q_{\mathsf{nxt(a)}} = \mathrm{NXT}_{\nabla P}(\mathsf{ctr(a)}, \mathsf{nxt(a)}, \overline{\mathsf{a}})$ associated with $\mathsf{a} \in \mathrm{Hb}(P)$. Now $\mathrm{head}(Q_{\mathsf{nxt(a)}}) = \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt(a)})$. The stability of N implies the local stability of N w.r.t. $Q_{\mathsf{nxt(a)}}$ by Theorem 6.22. Thus $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \mathrm{LM}(Q^N_{\mathsf{nxt(a)}})$. In analogy to the first item above, two cases arise depending on the membership of a in M. If $\mathsf{a} \in M$, then $\overline{\mathsf{a}} \notin N$ by (9.12). Moreover, we know by the first item that $N \cap \mathrm{head}(Q_{\mathsf{ctr(a)}}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr(a)}, \#\mathsf{a})$. Thus we may conclude by Lemma 9.5 that $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt(a)}, \#\mathsf{a} + 1 \bmod 2^{\nabla P})$. But if $\mathsf{a} \notin M$, then $\overline{\mathsf{a}} \in N$ by (9.12) and we obtain $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \emptyset$ by Lemma 9.5. Thus $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}})$ coincides with the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$ in both cases.

3. The case of a rule $r \in P$ with $\mathrm{body}^+(r) \neq \emptyset$ follows. The respective subprogram $Q_{\mathsf{ctr}(r)} = \mathrm{SEL}_{\nabla P}(\mathsf{ctr}(r), \overline{\mathsf{bt}(r)})$, which is included in $\mathrm{Tr}_{\mathrm{AT}}(P)$, is handled analogously to $Q_{\mathsf{ctr(a)}}$ in the first item. The only notable difference is that the controlling atom $\overline{\mathsf{bt}(r)}$ is governed by (9.13). This relationship gives the crucial link to the definitions of $\#r$ and $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$ in Definitions 6.15 and 6.18, respectively.

4. The last subprogram that needs our attention is the one associated with a rule $r \in P$ with $\mathrm{body}^+(r) = \emptyset$: let $Q_{\mathsf{ctr}(r)} = \mathrm{FIX}_{\nabla P}(\mathsf{ctr}(r), 1, \overline{\mathsf{bt}(r)})$. Now $\mathrm{head}(Q_{\mathsf{ctr}(r)}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), 1)$ and since N is locally stable w.r.t. $Q_{\mathsf{ctr}(r)}$, it holds that $N \cap \mathrm{head}(Q_{\mathsf{ctr}(r)}) = \mathrm{LM}(Q^N_{\mathsf{ctr}(r)})$. Two cases arise. If $\overline{\mathsf{bt}(r)} \notin N$, then $N \cap \mathrm{head}(Q_{\mathsf{ctr}(r)}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), 1)$ by Lemma 9.6 and $r \in \mathrm{SR}(P, M)$ by (9.13). Thus $\#r = 1$ by the definition of $\#r$ and $N \cap \mathrm{head}(Q_{\mathsf{ctr}(r)})$ complies with the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$. On the other hand, if $\overline{\mathsf{bt}(r)} \in N$, then $N \cap \mathrm{head}(Q_{\mathsf{ctr}(r)}) = \emptyset$ follows by Lemma 9.6. This is compatible with the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$, as $r \notin \mathrm{SR}(P, M)$ by (9.13). Hence N is locally stable w.r.t. $Q_{\mathsf{ctr}(r)}$.

Thus we have covered all subprograms of $\mathrm{Tr}_{\mathrm{CTR}}(P)$ contributing to the set $\mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$. In each case, we obtained a perfect compatibility w.r.t. $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$. Hence $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P)) = \mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$. □

9.8 Correctness of $\mathrm{Tr}_{\mathrm{MAX}}(P)$

Proposition 9.19 *Let P be a normal program. If M is a stable model of P, $\#$ the corresponding level numbering w.r.t. M, and $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$, then $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P)) = \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#) = \mathrm{LM}(\mathrm{Tr}_{\mathrm{MAX}}(P)^N)$.*

PROOF. Let M be a stable model of P and $\#$ the corresponding level numbering w.r.t. M, as implied by Theorem 5.8. Let us note that $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P)) = \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$ is implied directly by the definition of N and Lemma 9.13. The translation $\mathrm{Tr}_{\mathrm{MAX}}(P)$ given in Definition 6.10 consists of several subprograms leading to a case analysis. In each case, we establish the local stability of N w.r.t. the subprogram in question.

Then let $r \in P$ such that $\mathrm{body}^+(r) \neq \emptyset$. Subprograms of the translation $\mathrm{Tr}_{\mathrm{MAX}}(r, \nabla P)$ from Definition 6.10 are analyzed next.

1. Let $Q_{\mathsf{nxt(a)}} = \mathrm{LT}_{\nabla P}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}), \overline{\mathsf{bt}(r)})$ associated with an atom $\mathsf{a} \in \mathrm{body}^+(r)$. Two cases arise. If $r \in \mathrm{SR}(P, M)$, then $\mathsf{a} \in M$, as $M \models \mathrm{body}(r)$. Thus $\#r$ and $\#\mathsf{a}$ are well-defined, as $\#$ is a level numbering w.r.t. M. Moreover, $\#\mathsf{a} + 1 \bmod 2^{\nabla P} = \#\mathsf{a} + 1$ by Proposition 6.6. The atom $\overline{\mathsf{bt}(r)} \notin N$ by the definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. It follows by the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ that $N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r)) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#r)$ and $N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt}(\mathsf{a})) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#\mathsf{a} + 1)$. Also, we have $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \mathrm{AT}^{\mathrm{lt}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{nxt}(\mathsf{a}), \#\mathsf{a} + 1)$ by the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$. Consequently, Lemma 9.8 implies that $\mathrm{LM}(Q^N_{\mathsf{nxt(a)}}) = \mathrm{AT}^{\mathrm{lt}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{nxt}(\mathsf{a}), \#\mathsf{a} + 1)$.

   On the other hand, if $r \notin \mathrm{SR}(P, M)$ holds, then $\overline{\mathsf{bt}(r)} \in N$ by the definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. Then the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$ implies that $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \emptyset$, as $r \notin \mathrm{SR}(P, M)$. In accordance to this, we obtain $\mathrm{LM}(Q^N_{\mathsf{nxt(a)}}) = \emptyset$ by Lemma 9.8. To conclude the case analysis above, N is locally stable w.r.t. the subprogram $Q_{\mathsf{nxt(a)}}$.

2. The subprogram $R_{\mathsf{nxt(a)}} = \mathrm{EQ}_{\nabla P}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}), \overline{\mathsf{bt}(r)})$ associated with $\mathsf{a} \in \mathrm{body}^+(r)$ is covered analogously to $Q_{\mathsf{nxt(a)}}$, but we have to apply Lemma 9.11 rather than Lemma 9.8. Then $N \cap \mathrm{head}(R_{\mathsf{nxt(a)}}) = \mathrm{LM}(R^N_{\mathsf{nxt(a)}})$ which indicates that N is locally stable w.r.t. $R_{\mathsf{ctr}(r)}$.

3. Let

   $$Q_{\mathsf{max}(r)} = \{\mathsf{max}(r) \leftarrow {\sim}\overline{\mathsf{bt}(r)}, {\sim}\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))} \mid \mathsf{a} \in \mathrm{body}^+(r)\}.$$

   If $r \in \mathrm{SR}(P, M)$ holds, then $\overline{\mathsf{bt}(r)} \notin N$ follows by the definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. Then the rule $\mathsf{max}(r) \leftarrow$ belongs to the reduct $Q^N_{\mathsf{max}(r)}$

   $$\iff \exists \mathsf{a} \in \mathrm{body}^+(r) \text{ such that } \overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))} \notin N$$

   $\iff \exists \mathsf{a} \in \mathrm{body}^+(r)$ such that $\#r = \#\mathsf{a} + 1$ holds by the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$. The last is vacuously true by Theorem 5.8, as M is a supported model of P and $\#$ is a level numbering w.r.t. M. It follows by Lemma 4.9 that $\mathrm{LM}(Q^N_{\mathsf{max}(r)}) = \{\mathsf{max}(r)\}$ which equals to $N \cap \mathrm{head}(Q_{\mathsf{max}(r)})$ by the definition of the set $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$.

   If $r \notin \mathrm{SR}(P, M)$ holds, then $\overline{\mathsf{bt}(r)} \in N$ follows by the definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. The structure of $Q_{\mathsf{max}(r)}$ implies that $Q^N_{\mathsf{max}(r)} = \emptyset$ is this case. Thus $N \cap \mathrm{head}(Q_{\mathsf{max}(r)}) = \emptyset$, as dictated by the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$, equals to $\mathrm{LM}(Q^N_{\mathsf{max}(r)}) = \emptyset$. Hence N is locally stable w.r.t. $Q_{\mathsf{max}(r)}$.

Finally, let us consider the set of constraints $Q_{\mathsf{x}} =$

$$\bigcup_{r \in P \text{ and } \mathsf{a} \in \mathrm{body}^+(r)} \{\mathsf{x} \leftarrow {\sim}\mathsf{x}, {\sim}\overline{\mathsf{bt}(r)}, {\sim}\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))_1}\} \cup \bigcup_{r \in P \text{ and } \mathrm{body}^+(r) \neq \emptyset} \{\mathsf{x} \leftarrow {\sim}\mathsf{x}, {\sim}\overline{\mathsf{bt}(r)}, {\sim}\mathsf{max}(r)\}. \tag{9.14}$$

Note that $\mathsf{x} \notin N$ by definition and we would like to establish that $\mathsf{x} \leftarrow$ does not belong Q^N_{x}, which could result for two reasons.

1. Suppose that $\overline{\mathsf{bt}(r)} \notin N$ and $\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))_1} \notin N$ hold for some rule $r \in P$ such that $\mathrm{body}^+(r) \neq \emptyset$ and an atom $\mathsf{a} \in \mathrm{body}^+(r)$. The definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ implies that $r \in \mathrm{SR}(P, M)$, as $\overline{\mathsf{bt}(r)} \notin N$. Then the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$ implies that $\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))_1 \in N$ and $\#r < \#\mathsf{a} + 1$ for an atom $\mathsf{a} \in \mathrm{body}^+(r)$. A contradiction, as $r \in \mathrm{SR}(P, M)$ and $\#$ is a level numbering w.r.t. M.

2. Suppose that $\overline{\mathsf{bt}(r)} \notin N$ and $\mathsf{max}(r) \notin N$ for some rule $r \in P$ such that $\mathrm{body}^+(r) \neq \emptyset$. Then $r \in \mathrm{SR}(P, M)$ follows as above, and $\mathsf{max}(r) \in N$ follows by the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$, a contradiction.

It follows that $Q^N_{\mathsf{x}} = \emptyset$ so that $\mathrm{LM}(Q^N_{\mathsf{x}}) = \emptyset$ by Lemma 4.9. Thus $N \cap \mathrm{head}(Q_{\mathsf{x}}) = \mathrm{LM}(Q^N_{\mathsf{x}})$, which means that N is locally stable w.r.t. Q_{x}. To conclude, we have established that N is locally stable w.r.t. each subprogram of $\mathrm{Tr}_{\mathrm{MAX}}(P)$ and by Theorem 6.22, N is locally stable w.r.t. $\mathrm{Tr}_{\mathrm{MAX}}(P)$, too. □

Proposition 9.20 *Let P be a normal program. If N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, then $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P)) = \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$ for $M = N \cap \mathrm{Hb}(P)$ and the function $\# : M \cup \mathrm{SR}(P, M) \to \{0, \ldots, 2^{\nabla P} - 1\}$ from Definition 6.18.*

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, and define M and $\#$ as above. It follows by Theorem 6.22 that N is locally stable w.r.t. $\mathrm{Tr}_{\mathrm{MAX}}(P) \subseteq \mathrm{Tr}_{\mathrm{AT}}(P)$. Moreover, the relationships (9.12) and (9.13) pointed out in the proof of Proposition 9.18 are valid in this proof as well. Moreover, recall the translation $\mathrm{Tr}_{\mathrm{MAX}}(P)$ from Definition 6.10. In the sequel, we analyze subprograms that partition $\mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$ suitably. Note that Theorem 6.22 implies that N is locally stable w.r.t. each subprogram.

Recall the subprogram Q_{x} from (9.14) for which $\mathrm{head}(Q_{\mathsf{x}}) = \{\mathsf{x}\}$. Let us then assume that $\mathsf{x} \in N$. Since N is locally stable w.r.t. $\mathrm{Tr}_{\mathrm{MAX}}(P)$, it is locally stable w.r.t. Q_{x}, too. Thus $\mathsf{x} \in \mathrm{LM}(Q^N_{\mathsf{x}})$ so that $\mathsf{x} \leftarrow$ must belong to Q^N_{x}. Since $\sim\mathsf{x}$ appears in the body of each rule in Q_{x} this implies that $\mathsf{x} \notin N$, a contradiction. Hence $\mathsf{x} \notin N$. Accordingly, it holds that $\mathsf{x} \notin \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$ by the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$. Subprograms which are associated with rules $r \in P$ such that $\mathrm{body}^+(r) \neq \emptyset$ are analyzed next.

1. Let $Q_{\mathsf{nxt}(\mathsf{a})} = \mathrm{LT}_{\nabla P}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}), \overline{\mathsf{bt}(r)})$ associated with an atom $\mathsf{a} \in \mathrm{body}^+(r)$. Since N is locally stable w.r.t. $Q_{\mathsf{nxt}(\mathsf{a})}$, it holds that $N \cap \mathrm{head}(Q_{\mathsf{nxt}(\mathsf{a})}) = \mathrm{LM}(Q^N_{\mathsf{nxt}(\mathsf{a})})$. There are two cases to consider. First, if $\overline{\mathsf{bt}(r)} \notin N$, then $r \in \mathrm{SR}(P, M)$ by (9.13). Proposition 9.18 implies

$$(9.15) \quad \begin{cases} N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r)) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#r) \text{ and} \\ N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt}(\mathsf{a})) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt}(\mathsf{a}), \#\mathsf{a} + 1 \bmod 2^{\nabla P}). \end{cases}$$

Then $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \mathrm{AT}^{\mathrm{lt}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{nxt}(\mathsf{a}), \#\mathsf{a}+1 \bmod 2^{\nabla P})$ follows by Lemma 9.9. This is fully compatible with the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$. Second, if $\overline{\mathsf{bt}(r)} \in N$, we have $r \notin \mathrm{SR}(P, M)$ by (9.13). In addition to this, Lemma 9.9 implies that $N \cap \mathrm{head}(Q_{\mathsf{nxt(a)}}) = \emptyset$, in harmony with the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$.

2. Then consider $R_{\mathsf{nxt(a)}} = \mathrm{EQ}_{\nabla P}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}), \overline{\mathsf{bt}(r)})$ associated with $\mathsf{a} \in \mathrm{body}^+(r)$. This subprogram is handled analogously to the one above — starting from the local stability of N w.r.t. it. Like above, $\overline{\mathsf{bt}(r)} \notin N$ implies $r \in \mathrm{SR}(P, M)$ and (9.15). Then the equality $N \cap \mathrm{head}(R_{\mathsf{nxt(a)}}) = \mathrm{AT}^{\mathrm{eq}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{nxt}(\mathsf{a}), \#\mathsf{a}+1 \bmod 2^{\nabla P})$ follows by Lemma 9.12. Similarly, $\overline{\mathsf{bt}(r)} \in N$ implies $r \in \mathrm{SR}(P, M)$ and $N \cap \mathrm{head}(R_{\mathsf{nxt(a)}}) = \emptyset$ by Lemma 9.12. Hence the structure of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$ is respected.

3. Let

$$Q_{\mathsf{max}(r)} = \{\mathsf{max}(r) \leftarrow \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))} \mid \mathsf{a} \in \mathrm{body}^+(r)\}.$$

Since N is locally stable w.r.t. $Q_{\mathsf{max}(r)}$, it follows that $N \cap \mathrm{head}(Q_{\mathsf{max}(r)})$ equals to $\mathrm{LM}(Q^N_{\mathsf{max}(r)})$. Again, two cases arise. If $\overline{\mathsf{bt}(r)} \notin N$, then $r \in \mathrm{SR}(P, M)$ by (9.13). Let us then assume $\mathsf{max}(r) \notin N$. Since the rule $\mathsf{x} \leftarrow \sim\mathsf{x}, \sim\overline{\mathsf{bt}(r)}, \sim\mathsf{max}(r)$ is included in Q_{x}, we obtain that $\mathsf{x} \leftarrow$ is included in Q^N_{x}. Thus $\mathsf{x} \in N$, a contradiction. Hence $\mathsf{max}(r) \in N$ is necessarily the case. Recall that $\mathsf{max}(r) \in \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$, too. On the other hand, if $\overline{\mathsf{bt}(r)} \in N$, then $r \notin \mathrm{SR}(P, M)$ by (9.13). It follows that $Q^N_{\mathsf{max}(r)} = \emptyset$ so that $\mathsf{max}(r) \notin \mathrm{LM}(Q^N_{\mathsf{max}(r)})$. Thus we have a perfect match with the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$ which excludes $\mathsf{max}(r)$ when $r \notin \mathrm{SR}(P, M)$.

We have now covered all sets of atoms that constitute $\mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$. It follows that $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P)) = \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$. □

Proposition 9.21 *Let P be a normal program. If N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, $M = N \cap \mathrm{Hb}(P)$ and $\#$ from Definition 6.18, then*

$$\#r = \begin{cases} \max\{\#\mathsf{a}+1 \bmod 2^{\nabla P} \mid \mathsf{a} \in \mathrm{body}^+(r)\}, & \textit{if } \mathrm{body}^+(r) \neq \emptyset. \\ 1, & \textit{otherwise.} \end{cases} \tag{9.16}$$

holds for every $r \in \mathrm{SR}(P, M)$.

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$ and define M as above and $\#$ by Definition 6.18. Then let us consider any $r \in P$ such that $r \in \mathrm{SR}(P, M)$ for which $\#r$ is well-defined. As established in Proposition 9.18, this particular value is held by $\mathsf{ctr}(r)$ in binary. Then (9.13) implies that $\overline{\mathsf{bt}(r)} \notin N$. Two cases arise depending whether $\mathrm{body}^+(r) = \emptyset$ or not.

If $\mathrm{body}^+(r) = \emptyset$, then $N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r)) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), 1)$ by Proposition 9.18. Thus $\#r = \mathrm{val}_{\nabla P}(\mathsf{ctr}(r), N) = 1$, as insisted by Definition 6.18.

If $\mathrm{body}^+(r) \neq \emptyset$, there is at least one atom $\mathsf{a} \in \mathrm{body}^+(r)$. Let us assume that $\#r < \#\mathsf{a}+1 \bmod 2^{\nabla P}$. Recall that the value $\#\mathsf{a}+1 \bmod 2^{\nabla P}$ is held by

the counter $\mathsf{nxt}(\mathsf{a})$, i.e. $N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt}(\mathsf{a})) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{nxt}(\mathsf{a}), \#\mathsf{a}+1 \bmod 2^{\nabla P})$ by Proposition 9.18. Since $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P)) = \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$ by Proposition 9.20, we obtain $\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))_1} \notin N$. Then recall that the rule $\mathsf{x} \leftarrow {\sim}\mathsf{x}, {\sim}\overline{\mathsf{bt}(r)}, {\sim}\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))_1}$ included in $\mathrm{Tr}_{\mathrm{MAX}}(P)$. It follows that $\mathsf{x} \leftarrow$ is included in $\mathrm{Tr}_{\mathrm{MAX}}(P)^N$ and furthermore $\mathsf{x} \in N$, a contradiction. Thus we conclude that $\#r \geq \#\mathsf{a} + 1 \bmod 2^{\nabla P}$ actually holds for all $\mathsf{a} \in \mathrm{body}^+(r)$.

On the other hand, we established $\mathsf{max}(r) \in N$ in the proof of Proposition 9.20 where the program $Q_{\mathsf{max}(r)}$ was analyzed. Since N is locally stable w.r.t. $Q_{\mathsf{max}(r)}$, it follows that $\mathsf{max}(r) \in \mathrm{LM}(Q^N_{\mathsf{max}(r)})$. Then Lemma 4.9 implies that $\mathsf{max}(r) \leftarrow$ belongs to $Q^N_{\mathsf{max}(r)}$. This is possible only if there is $\mathsf{a} \in \mathrm{body}^+(r)$ such that $\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))} \notin N$. It follows by Proposition 9.20 that $\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))} \notin \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$, or equivalently, it holds that $\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a})) \in \mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$. Then the definition of $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$ implies that $\#r = \#\mathsf{a} + 1 \bmod 2^{\nabla P}$. To conclude, we have established that

$$\#r = \max\{\#\mathsf{a} + 1 \bmod 2^{\nabla P} \mid \mathsf{a} \in \mathrm{body}^+(r)\}.$$

□

9.9 Correctness of $\mathrm{Tr}_{\mathrm{MIN}}(P)$

Proposition 9.22 *Let P be a normal program. If M is a stable model of P, $\#$ the corresponding level numbering w.r.t. M, and $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$, then $\mathrm{LM}(\mathrm{Tr}_{\mathrm{MIN}}(P)^N) = \mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$.*

PROOF. Let M be a stable model of P and $\#$ the corresponding level numbering, as implied by Theorem 5.8. We note that $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P)) = \mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ holds directly by the definition of N and Lemma 9.13. The translation $\mathrm{Tr}_{\mathrm{MIN}}(P)$ given in Definition 6.11 consists of multiple subprograms leading to a case analysis. In each case, we establish the local stability of N w.r.t. the subprogram in question.

Given a rule $r \in P$, the subprograms of $\mathrm{Tr}_{\mathrm{MIN}}(r, \nabla P)$ from Definition 6.11 are addressed next. Suppose that $\mathrm{head}(r) = \mathsf{a}$.

1. Let $Q_{\mathsf{ctr}(\mathsf{a})} = \mathrm{LT}_{\nabla P}(\mathsf{ctr}(r), \mathsf{ctr}(\mathsf{a}), \overline{\mathsf{bt}(r)})$. Two cases arise depending on the status of r. If $r \in \mathrm{SR}(P, M)$, then $\mathsf{a} \in M$, as $M \models \mathrm{body}(r)$ and $M \models r$. Thus $\#r$ and $\#\mathsf{a}$ are well-defined, as $\#$ is a level numbering w.r.t. M. The atom $\overline{\mathsf{bt}(r)} \notin N$ by the definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. It follows by the definition of $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$ that $N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r)) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#r)$ as well as that $N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(\mathsf{a})) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), \#\mathsf{a})$. Also, we have

$$N \cap \mathrm{head}(Q_{\mathsf{ctr}(\mathsf{a})}) = \mathrm{AT}^{\mathrm{lt}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{ctr}(\mathsf{a}), \#\mathsf{a})$$

by the definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$. Thus

$$\mathrm{LM}(Q^N_{\mathsf{ctr}(\mathsf{a})}) = \mathrm{AT}^{\mathrm{lt}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{ctr}(\mathsf{a}), \#\mathsf{a})$$

follows by Lemma 9.8.

On the other hand, if $r \notin \mathrm{SR}(P, M)$ holds, then $\overline{\mathsf{bt}(r)} \in N$ by the definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$. The definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$ implies that $N \cap \mathrm{head}(Q_{\mathsf{ctr}(\mathsf{a})}) = \emptyset$, as $r \notin \mathrm{SR}(P, M)$. In accordance to this, we obtain $\mathrm{LM}(Q^N_{\mathsf{ctr}(\mathsf{a})}) = \emptyset$ by Lemma 9.8. To conclude the preceding case analysis, N is locally stable w.r.t. $Q_{\mathsf{ctr}(\mathsf{a})}$.

2. The subprogram $R_{\mathsf{ctr}(\mathsf{a})} = \mathrm{EQ}_{\nabla P}(\mathsf{ctr}(r), \mathsf{ctr}(\mathsf{a}), \overline{\mathsf{bt}(r)})$ is covered analogously to $Q_{\mathsf{ctr}(\mathsf{a})}$, but we have to apply Lemma 9.11 in this case rather than Lemma 9.8. Then $N \cap \mathrm{head}(R_{\mathsf{ctr}(\mathsf{a})}) = \mathrm{LM}(R^N_{\mathsf{ctr}(\mathsf{a})})$ which indicates that N is locally stable w.r.t. the subprogram $R_{\mathsf{ctr}(\mathsf{a})}$.

The following subprogram is associated with any atom $\mathsf{a} \in \mathrm{head}(P)$. Let $Q_{\mathsf{min}(\mathsf{a})} = \{\mathsf{min}(\mathsf{a}) \leftarrow \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr}(\mathsf{a}))} \mid r \in \mathrm{Def}_P(\mathsf{a})\}$. Note that $\mathrm{head}(Q_{\mathsf{min}(\mathsf{a})}) = \{\mathsf{min}(\mathsf{a})\}$. It follows that $\mathsf{min}(\mathsf{a}) \in N \iff \mathsf{a} \in M$ by the definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P)) \iff \exists r \in \mathrm{Def}_P(\mathsf{a})$ such that $r \in \mathrm{SR}(P, M)$, as M is a supported model of P, $\iff$ $\exists r \in \mathrm{Def}_P(\mathsf{a})$ such that $r \in \mathrm{SR}(P, M)$ and $\#r = \#\mathsf{a}$ by Definition 5.2, as $\#$ is a level numbering w.r.t. M, $\iff \exists r \in \mathrm{Def}_P(\mathsf{a})$ such that $\overline{\mathsf{bt}(r)} \notin N$ and $\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr}(\mathsf{a}))} \notin N$ by the definitions of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ and $\mathrm{Ext}_{\mathrm{MIN}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P)) \iff$ the rule $\mathsf{min}(\mathsf{a}) \leftarrow$ belongs to the reduct $Q^N_{\mathsf{min}(\mathsf{a})} \iff \mathsf{min}(\mathsf{a}) \in \mathrm{LM}(Q^N_{\mathsf{min}(\mathsf{a})})$. Hence N is locally stable w.r.t. $Q_{\mathsf{min}(\mathsf{a})}$.

Finally, let us consider the set of constraints

$$Q_{\mathsf{y}} = \bigcup_{r \in P} \{\mathsf{y} \leftarrow \sim\mathsf{y}, \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r)))_1}\} \cup \bigcup_{\mathsf{a} \in \mathrm{Hb}(P)} \{\mathsf{y} \leftarrow \sim\mathsf{y}, \sim\overline{\mathsf{a}}, \sim\mathsf{min}(\mathsf{a})\}. \tag{9.17}$$

Note that $\mathsf{y} \notin N$ by definition and we would like to establish that $\mathsf{y} \leftarrow$ does not belong Q^N_{y}, which might result for two reasons.

1. Let $\overline{\mathsf{bt}(r)} \notin N$ and $\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{ctr}(\mathrm{head}(r)))_1} \notin N$ hold for some $r \in P$. The definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ implies that $r \in \mathrm{SR}(P, M)$, as $\overline{\mathsf{bt}(r)} \notin N$. Note that $r \in \mathrm{SR}(P, M)$ implies that $\mathrm{head}(r) = \mathsf{a} \in M$, as $M \models r$. Then $\#r < \#\mathsf{a}$ follows by the definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$. A contradiction by Definition 5.2, as $\mathsf{a} \in M$ and $\#$ is a level numbering w.r.t. M.

2. Suppose that $\overline{\mathsf{a}} \notin N$ and $\mathsf{min}(\mathsf{a}) \notin N$ hold for some atom $\mathsf{a} \in \mathrm{Hb}(P)$. Then the definition of $\mathrm{Ext}_{\mathrm{SUPP}}(P, M) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$ implies $\mathsf{a} \in M$. Moreover, $\mathsf{min}(\mathsf{a}) \in N$ follows by the definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#) = N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$, a contradiction.

It follows that $Q^N_{\mathsf{y}} = \emptyset$ so that $\mathrm{LM}(Q^N_{\mathsf{y}}) = \emptyset$ by Lemma 4.9. Thus $N \cap \mathrm{head}(Q_{\mathsf{y}}) = \mathrm{LM}(Q^N_{\mathsf{y}})$, which means that N is locally stable w.r.t. Q_{y}. To conclude, we have established that N is locally stable w.r.t. each subprogram of $\mathrm{Tr}_{\mathrm{MIN}}(P)$. Hence N is locally stable w.r.t. $\mathrm{Tr}_{\mathrm{MIN}}(P)$ by Theorem 6.22. □

Proposition 9.23 *Let P be a normal program. If N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, then $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P)) = \mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ for $M = N \cap \mathrm{Hb}(P)$ and the function $\# : M \cup \mathrm{SR}(P, M) \to \{0, \ldots, 2^{\nabla P} - 1\}$ from Definition 6.18.*

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, and define M and $\#$ as above. It follows by Theorem 6.22 that N is locally stable w.r.t. $\mathrm{Tr}_{\mathrm{MIN}}(P) \subseteq \mathrm{Tr}_{\mathrm{AT}}(P)$. Moreover, the relationships (9.12) and (9.13) pointed out in the proof of Proposition 9.18 are valid in this proof as well. Let us recall the translation $\mathrm{Tr}_{\mathrm{MIN}}(P)$ and the set $\mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$ from Definition 6.10. In the sequel, we analyze subprograms $Q \subseteq \mathrm{Tr}_{\mathrm{MIN}}(P)$ which partition the set $\mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$ suitably such that Theorem 6.22 implies that N is locally stable w.r.t. each subprogram Q being analyzed.

Recall the program Q_{y} from (9.17) for which $\mathrm{head}(Q_{\mathsf{y}}) = \{\mathsf{y}\}$. Let us then assume that $\mathsf{y} \in N$. Since N is locally stable w.r.t. $\mathrm{Tr}_{\mathrm{MIN}}(P)$, it is locally stable w.r.t. Q_{y}, too. Thus $\mathsf{y} \in \mathrm{LM}(Q_{\mathsf{y}}^N)$ so that $\mathsf{y} \leftarrow$ must belong to Q_{y}^N. Since $\sim\mathsf{y}$ appears in the body of each rule in Q_{y} this implies that $\mathsf{y} \notin N$, a contradiction. Hence $\mathsf{y} \notin N$. Accordingly, we have $\mathsf{y} \notin \mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ by the definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$.

Let us then address the subprograms of $\mathrm{Tr}_{\mathrm{MIN}}(r, \nabla P)$ which are associated with a rule $r \in P$. Let $\mathrm{head}(r) = \mathsf{a}$.

1. Let $Q_{\mathsf{ctr(a)}} = \mathrm{LT}_{\nabla P}(\mathsf{ctr}(r), \mathsf{ctr(a)}, \overline{\mathsf{bt}(r)})$. Since N is locally stable w.r.t. $Q_{\mathsf{ctr(a)}}$, it holds that $N \cap \mathrm{head}(Q_{\mathsf{ctr(a)}}) = \mathrm{LM}(Q_{\mathsf{ctr(a)}}^N)$. There are two cases to consider. First, if $\overline{\mathsf{bt}(r)} \notin N$, then $r \in \mathrm{SR}(P, M)$ by (9.13) and Proposition 9.18 implies the following:

$$(9.18) \qquad \begin{cases} N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r)) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#r) \text{ and} \\ N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr(a)}) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr(a)}, \#\mathsf{a}). \end{cases}$$

   Then $N \cap \mathrm{head}(Q_{\mathsf{ctr(a)}}) = \mathrm{AT}^{\mathrm{lt}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{nxt(a)}, \#\mathsf{a})$ is obtained by Lemma 9.9. This conforms to the definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$. Second, if $\overline{\mathsf{bt}(r)} \in N$, we have $r \notin \mathrm{SR}(P, M)$ by (9.13). On the other hand, Lemma 9.9 implies that $N \cap \mathrm{head}(Q_{\mathsf{ctr(a)}}) = \emptyset$, in harmony with the definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$.

2. Then consider $R_{\mathsf{ctr(a)}} = \mathrm{EQ}_{\nabla P}(\mathsf{ctr}(r), \mathsf{ctr(a)}, \overline{\mathsf{bt}(r)})$. This subprogram is handled analogously to the one above — starting from the local stability of N w.r.t. $R_{\mathsf{ctr(a)}}$. Like above, $\overline{\mathsf{bt}(r)} \notin N$ implies $r \in \mathrm{SR}(P, M)$ and (9.18). Then $N \cap \mathrm{head}(R_{\mathsf{ctr(a)}}) = \mathrm{AT}^{\mathrm{eq}}_{\nabla P}(\mathsf{ctr}(r), \#r, \mathsf{ctr(a)}, \#\mathsf{a})$ follows by Lemma 9.12. Quite similarly, $\overline{\mathsf{bt}(r)} \in N$ implies $r \notin \mathrm{SR}(P, M)$ and $N \cap \mathrm{head}(R_{\mathsf{ctr(a)}}) = \emptyset$ by Lemma 9.12. To conclude, the structure of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ is respected.

Finally, given any atom $\mathsf{a} \in \mathrm{head}(P)$, let us define

$$Q_{\mathsf{min(a)}} = \{\mathsf{min(a)} \leftarrow \sim\overline{\mathsf{bt}(r)}, \sim\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr(a)})} \mid r \in \mathrm{Def}_P(\mathsf{a})\}.$$

Since N is locally stable w.r.t. $Q_{\mathsf{min(a)}}$, it holds that $N \cap \mathrm{head}(Q_{\mathsf{min(a)}}) = \mathrm{LM}(Q_{\mathsf{min(a)}}^N)$. Two cases arise. If $\overline{\mathsf{a}} \notin N$, then $\mathsf{a} \in M$ by (9.12). Let us then assume $\mathsf{min(a)} \notin N$. Since the rule $\mathsf{y} \leftarrow \sim\mathsf{y}, \sim\overline{\mathsf{a}}, \sim\mathsf{min(a)}$ is included in Q_{y},

we obtain that $\mathsf{y} \leftarrow$ is included in Q^N_{y}. Thus $\mathsf{y} \in N$, a contradiction. Hence $\mathsf{min}(\mathsf{a}) \in N$ must hold. Recall that $\mathsf{min}(\mathsf{a}) \in \mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$, too. On the other hand, if $\overline{\mathsf{a}} \in N$, then $\mathsf{a} \notin M$ by (9.12). Since M is a supported model of P, it follows that $r \notin \mathrm{SR}(P, M)$ holds for every $r \in \mathrm{Def}_P(\mathsf{a})$. Thus $\overline{\mathsf{bt}(r)} \in N$ for all $r \in \mathrm{Def}_P(\mathsf{a})$ by (9.13). It follows that $Q^N_{\mathsf{min}(\mathsf{a})} = \emptyset$ so that $\mathsf{min}(\mathsf{a}) \notin \mathrm{LM}(Q^N_{\mathsf{min}(\mathsf{a})}) = \emptyset$. This matches with the definition of $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ which excludes $\mathsf{min}(\mathsf{a})$ when $\mathsf{a} \notin M$. To conclude, we have established that $N \cap \mathrm{head}(Q_{\mathsf{min}(\mathsf{a})})$ is fully compatible with $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$.

In this way, we have covered all subprograms contributing a partition to $\mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$. Thus $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P)) = \mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$. □

Proposition 9.24 *Let P be a normal program. If N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$, $M = N \cap \mathrm{Hb}(P)$ and the function $\#$ from Definition 6.18, then*

$$\#\mathsf{a} = \min\{\#r \mid r \in \mathrm{SR}(P, M) \text{ and } \mathsf{a} = \mathrm{head}(r)\} \tag{9.19}$$

holds for every $\mathsf{a} \in M$.

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$. Then define M as above and the function $\#$ using Definition 6.18. It follows by Proposition 9.16 that M is a supported model of P. Then let us consider any $\mathsf{a} \in M$, for which $\overline{\mathsf{a}} \notin N$ holds by (9.12).

Let r be any rule $r \in \mathrm{Def}_P(\mathsf{a})$ such that $r \in \mathrm{SR}(P, M)$. Then both $\#\mathsf{a}$ and $\#r$ are well-defined by Definition 6.18. Let us then assume that $\#r < \#\mathsf{a}$. Recall that $N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r)) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(r), \#r)$ and $N \cap \mathrm{Hb}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(\mathsf{a})) = \mathrm{AT}^{\mathrm{ctr}}_{\nabla P}(\mathsf{ctr}(\mathsf{a}), \#\mathsf{a})$ by Proposition 9.18. Since the set $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$ equals to $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ by Proposition 9.23, it follows that $\overline{\mathsf{lt}(\mathsf{ctr}(r), \mathsf{ctr}(\mathsf{a}))_1} \notin N$. Then recall that the rule

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

included in $\mathrm{Tr}_{\mathrm{MIN}}(P)$. Now (9.13) implies that $\overline{\mathsf{bt}(r)} \notin N$. It follows that $\mathsf{y} \leftarrow$ is included in $\mathrm{Tr}_{\mathrm{MIN}}(P)^N$ and furthermore $\mathsf{y} \in N$. But this is a contradiction w.r.t. Proposition 9.23. Thus we may conclude that $\#r \geq \#\mathsf{a}$ actually holds for all $r \in \mathrm{Def}_P(\mathsf{a})$ such that $r \in \mathrm{SR}(P, M)$.

On the other hand, we established $\mathsf{min}(\mathsf{a}) \in N$ in the proof of Proposition 9.23, where the program $Q_{\mathsf{min}(\mathsf{a})}$ was analyzed. Since N is locally stable w.r.t. $Q_{\mathsf{min}(r)}$, it follows that $\mathsf{min}(\mathsf{a}) \in \mathrm{LM}(Q^N_{\mathsf{min}(\mathsf{a})})$. Then Lemma 4.9 implies that $\mathsf{min}(\mathsf{a}) \leftarrow$ belongs to $Q^N_{\mathsf{min}(\mathsf{a})}$. This is possible only if there is $r \in \mathrm{Def}_P(\mathsf{a})$ such that $\overline{\mathsf{bt}(r)} \notin N$ and $\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{nxt}(\mathsf{a}))} \notin N$. It follows that $r \in \mathrm{SR}(P, M)$ by (9.13). Moreover, Proposition 9.23 implies that $\overline{\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr}(\mathsf{a}))} \notin \mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$. Then the definition of the set $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ implies that $\mathsf{eq}(\mathsf{ctr}(r), \mathsf{ctr}(\mathsf{a})) \in \mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ which indicates that $\#r = \#\mathsf{a}$ is in fact the case. To conclude, we have established the claim regarding $\#\mathsf{a}$. □

9.10 Correctness of $\mathrm{Tr}_{\mathrm{AT}}(P)$

Proposition 9.25 *If M is a supported model of a normal program P and $\# : M \cup \mathrm{SR}(P, M) \rightarrow \{0, \ldots, 2^{\nabla P} - 1\}$ a function satisfying (9.16) and (9.19), then $\#$ is a level numbering w.r.t. M.*

PROOF. We should essentially establish the requirement imposed on $\#r$ in Definition 5.2 using (9.16) and (9.19). If $M = \emptyset$, then also $\mathrm{SR}(P, M) = \emptyset$, since $M \models P$, and $\#$ is trivially a level numbering w.r.t. M. Thus we assume $M \neq \emptyset$ and its implication $\mathrm{SR}(P, M) \neq \emptyset$ in the sequel. Let r be any rule from $\mathrm{SR}(P, M)$. By (9.16) the value $\#r$ is either 1 when $\mathrm{body}^+(r) = \emptyset$; or $\#r$ is the maximum among the values $\#\mathsf{b} + 1 \bmod 2^{\nabla P}$ where $\mathsf{b} \in \mathrm{body}^+(r)$ when $\mathrm{body}^+(r) \neq \emptyset$. In the latter case, the level number $\#\mathsf{b}$ is the predecessor of $\#r$ modulo $2^{\nabla P}$ for some $\mathsf{b} \in \mathrm{body}^+(r)$. Note that $\mathsf{b} \in M$, as $r \in \mathrm{SR}(P, M)$. Since M is a supported model of P, there is a rule $r' \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r') = \mathsf{b}$ and $\#\mathsf{b} = \#r'$ is the minimum (9.19) and the predecessor of $\#r$ modulo $2^{\nabla P}$; i.e. $\#r = \#r' + 1 \bmod 2^{\nabla P}$. To conclude our analysis so far,

1. either $\mathrm{body}^+(r) = \emptyset$ and $\#r = 1$; or
2. $\mathrm{body}^+(r) \neq \emptyset$ and there are $\mathsf{b} \in \mathrm{body}^+(r)$ and $r' \in \mathrm{SR}(P, M)$ such that $\mathrm{head}(r') = \mathsf{b} \in M$, $\#r' = \#\mathsf{b}$, and $\#r = \#r' + 1 \bmod 2^{\nabla P}$.

Since r was arbitrary in the preceding analysis, we can build a sequence of rules $r_1, \ldots, r_n$ from $\mathrm{SR}(P, M)$ such that $r_1 = r$, for all $i \in \{1, \ldots, n-1\}$, $\mathrm{body}^+(r_i) \neq \emptyset$ and $\#r_i = \#r_{i+1} + 1 \bmod 2^{\nabla P}$; and $\mathrm{body}^+(r_n) = \emptyset$. Suppose that such r_n cannot be found, i.e. the sequence $r_1, r_2, \ldots$ contains only rules r_i with $\mathrm{body}^+(r_i) = \emptyset$. Note that the sequence

$$\#\mathrm{head}(r_1), \#\mathrm{head}(r_2), \ldots$$

is decreasing modulo $2^{\nabla P}$ by construction and also identical to the sequence $\#r_1, \#r_2, \ldots$ after the first element. Since $\#$ is a function, we conclude that the head atoms $\mathrm{head}(r_2), \mathrm{head}(r_3), \ldots$ must be different from each other at least until the rule $r_{(2^{\nabla P}+1)}$, which is encountered after $2^{\nabla P}$ steps and the level number of which coincides with $\#r_1$ due to modulo arithmetics. As a consequence, there are $2^{\nabla P}$ different atoms in $\mathrm{Hb}(P)$. A contradiction, as $2^{\nabla P} \geq |\mathrm{Hb}(P)| + 2$. Hence the existence of r_n with $\mathrm{body}^+(r) = \emptyset$ is guaranteed. Given the range of $\#$, this implies that $\#r = n$ where $n < 2^{\nabla P}$. A further implication is that the level numbers $\#r$ assigned to rules $r \in \mathrm{SR}(P, M)$ are always greater than 0, because r was freely chosen above.

As a consequence, the level numbers $\#\mathsf{a}$ assigned to atoms $\mathsf{a} \in M$ are greater than 0 by (9.19). Let us then assume that $\#\mathsf{a} = 2^{\nabla P} - 1$ for some $\mathsf{a} \in M$. By (9.19) there is a rule $r \in \mathrm{SR}(P, M)$ such that $\#r = \#\mathsf{a} = 2^{\nabla P} - 1$ and we can construct a sequence of rules $r_1, \ldots, r_n$ as above, which starts from the rule $r_1 = r$. Since $\#r = 2^{\nabla P} - 1$, the length of the sequence must be $n = 2^{\nabla P} - 1$. Moreover, the values $\#r_1 = \#\mathrm{head}(r_1)$, $\#r_2 = \#\mathrm{head}(r_2)$, form a decreasing sequence. Since $\#$ is a function, there must be $2^{\nabla P} - 1$ different (head) atoms in $\mathrm{head}(P) \subseteq \mathrm{Hb}(P)$ as well as $\mathrm{Hb}(P)$. A contradiction, as $2^{\nabla P} - 1 \geq |\mathrm{Hb}(P)| + 2 - 1 = |\mathrm{Hb}(P)| + 1$. Thus $0 < \#\mathsf{a} < 2^{\nabla P} - 1$ for all $\mathsf{a} \in M$. Given this crucial piece of information and a rule $r \in \mathrm{SR}(P, M)$ with $\mathrm{body}^+(r) \neq \emptyset$, the first part of the equation (9.16) can be rewritten as

$$\begin{aligned} \#r &= \max\{\#\mathsf{b} + 1 \bmod 2^{\nabla P} \mid \mathsf{b} \in \mathrm{body}^+(r)\} \\ &= \max\{\#\mathsf{b} + 1 \mid \mathsf{b} \in \mathrm{body}^+(r)\} \\ &= \max\{\#\mathsf{b} \mid \mathsf{b} \in \mathrm{body}^+(r)\} + 1. \end{aligned}$$

To conclude, $\#$ is a level numbering w.r.t. M, as it is a function from $M \cup \mathrm{SR}(P, M)$ to $\mathbb{N}$ and the requirements of Definition 5.2 are met. □

Proposition 9.26 *Let P be a normal program. If M is a stable model of P and $\#$ is the corresponding level numbering w.r.t. M, then the interpretation $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$ is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$ such that $M = N \cap \mathrm{Hb}(P)$.*

PROOF. Let M be a stable model of P. The existence of the unique level numbering $\#$ w.r.t. M is implied by Theorem 5.8. Define N as the set of atoms $\mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$ given in Definition 6.16.

The translations $\mathrm{Tr}_{\mathrm{SUPP}}(P)$, $\mathrm{Tr}_{\mathrm{CTR}}(P)$, $\mathrm{Tr}_{\mathrm{MAX}}(P)$, and $\mathrm{Tr}_{\mathrm{MIN}}(P)$ which constitute $\mathrm{Tr}_{\mathrm{AT}}(P)$ have been designed so that $\mathrm{head}(\mathrm{Tr}_{\mathrm{AT}}(P))$ is partitioned by the respective sets of head atoms. Moreover, Propositions 9.14, 9.17, 9.19, and 9.22 imply that N is locally stable w.r.t. these partitions of $\mathrm{Tr}_{\mathrm{AT}}(P)$. It follows by Theorem 6.22 that N is a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$. Moreover, the projection $N \cap \mathrm{Hb}(P) = M$ by Definitions 6.14 and 6.16. □

Proposition 9.27 *Let P be a normal program. If N is a stable model of the translation $\mathrm{Tr}_{\mathrm{AT}}(P)$, then $M = N \cap \mathrm{Hb}(P)$ is a stable model of P, the function $\#$ from Definition 6.18 is a level numbering w.r.t. M and $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$.*

PROOF. Let N be a stable model of $\mathrm{Tr}_{\mathrm{AT}}(P)$ and define $M = N \cap \mathrm{Hb}(P)$ as above. Note that the set $\mathrm{SR}(P, M)$ is also well-defined, as M is. Thus we may define a function $\#$ — a level numbering candidate — using Definition 6.18. It follows by Proposition 9.16 that M is a supported model of P. Moreover, the function $\#$ meets the requirements for a level numbering w.r.t. M by Propositions 9.21, 9.24, and 9.25. Thus M is a stable model of P. Moreover, the projections $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P))$, $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P))$, $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P))$, and $N \cap \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$ equal to the sets $\mathrm{Ext}_{\mathrm{SUPP}}(P, M)$ (Proposition 9.15), $\mathrm{Ext}_{\mathrm{CTR}}(P, M, \#)$ (Proposition 9.18), $\mathrm{Ext}_{\mathrm{MAX}}(P, M, \#)$ (Proposition 9.20), and $\mathrm{Ext}_{\mathrm{MIN}}(P, M, \#)$ (Proposition 9.23), respectively. By Definition 6.16, the union of these sets is $\mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$. Since $N \subseteq \mathrm{head}(\mathrm{Tr}_{\mathrm{AT}}(P))$ and the disjoint union of

$$\mathrm{head}(\mathrm{Tr}_{\mathrm{SUPP}}(P)), \mathrm{head}(\mathrm{Tr}_{\mathrm{CTR}}(P)), \mathrm{head}(\mathrm{Tr}_{\mathrm{MAX}}(P)), \text{ and } \mathrm{head}(\mathrm{Tr}_{\mathrm{MIN}}(P))$$

coincides with $\mathrm{head}(\mathrm{Tr}_{\mathrm{AT}}(P))$, it follows that $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$. □

PROOF of Theorem 6.20. Let us first address the polynomiality of $\mathrm{Tr}_{\mathrm{AT}}$. Given a normal logic program P, it is obvious that the Herbrand base $\mathrm{Hb}(P)$ can be determined by performing a linear pass through the program. This is basically a linear-time operation, but the removal of duplicates adds a logarithmic factor in the worst case — recall that $\mathrm{Hb}(P)$ is a *set* of atoms rather than a *list* of atoms. Once $\mathrm{Hb}(P)$ is known, then $|\mathrm{Hb}(P)|$ and ∇P can be computed in linear/logarithmic time. Let us then consider each one of the translations $\mathrm{Tr}_{\mathrm{SUPP}}(P)$, $\mathrm{Tr}_{\mathrm{CTR}}(P)$, $\mathrm{Tr}_{\mathrm{MAX}}(P)$, and $\mathrm{Tr}_{\mathrm{MIN}}(P)$ separately.

1. Recall $\mathrm{Tr}_{\mathrm{SUPP}}(P)$ from Definition 6.8. Each atom $\mathsf{a} \in \mathrm{Hb}(P)$ is translated into a rule of five symbols; recall that separators count as one extra

symbol for each rule. A rule (2.1) consisting of $2n + 3m + 3$ symbols is translated into three rules consisting of $2n + 3m + 13$ symbols in total. Thus $||\mathrm{Tr}_{\mathrm{SUPP}}(P)||$ is $5 \times |\mathrm{Hb}(P)| + ||P|| + 10 \times |P|$ which is clearly linear in $||P||$, as $|\mathrm{Hb}(P)| < ||P||$ and $|P| < ||P||$. Furthermore, the translation can be formed in linear time by doing a linear pass through $\mathrm{Hb}(P)$ and P and producing the required rules.

2. Let us then consider the translation $\mathrm{Tr}_{\mathrm{CTR}}(P)$ from Definition 6.9, which uses the primitives from Table 2 as subprograms. By counting the symbols in the rules involved in these programs, we obtain the lengths $||\mathrm{SEL}_j(\mathsf{a}, \mathsf{c})|| = 16j$, $||\mathrm{NXT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})|| = 72j - 56$, and $||\mathrm{FIX}_j(\mathsf{a}, n, \mathsf{c})|| = 5j$ in general. The translation $\mathrm{Tr}_{\mathrm{CTR}}(P)$ concerns both atoms $\mathsf{a} \in \mathrm{Hb}(P)$ and rules $r \in P$. The translation of an atom a involves $16\nabla P + 72\nabla P - 56 = 88\nabla P - 56$ symbols whereas the translation of a rule r contains either $16\nabla P$ or $5\nabla P$ — depending on $\mathrm{body}^+(r)$. Therefore, $||\mathrm{Tr}_{\mathrm{CTR}}(P)||$ is bounded from above by the number $|\mathrm{Hb}(P)| \times (88\nabla P - 56) + |P| \times 16\nabla P$ which is linear in $||P|| \times \nabla P$, since $|\mathrm{Hb}(P)| < ||P||$ and $|P| < ||P||$. It is also obvious that $\mathrm{Tr}_{\mathrm{CTR}}(P)$ can be formed by doing a linear pass through $\mathrm{Hb}(P)$ and P. Thus $\mathrm{Tr}_{\mathrm{CTR}}(P)$ can be produced in time linear to $||P|| \times \nabla P$.

3. The translation $\mathrm{Tr}_{\mathrm{MAX}}(P)$ from Definition 6.10 is based on the primitives in Table 3. Generally speaking, the lengths of the programs therein are $||\mathrm{LT}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})|| = 47j - 28$ and $||\mathrm{EQ}_j(\mathsf{a}, \mathsf{b}, \mathsf{c})|| = 22j + 8$. The translation $\mathrm{Tr}_{\mathrm{MAX}}(P)$ deals basically with positive body atoms $\mathsf{a} \in \mathrm{body}^+(r)$ appearing in non-atomic rules r. Then $||\mathrm{Tr}_{\mathrm{MAX}}(r, \nabla P, \mathsf{a})||$ is $47\nabla P - 28 + 22\nabla P + 8 + 19 = 69\nabla P - 1$. As a consequence, $||\mathrm{Tr}_{\mathrm{MAX}}(P)||$ is bounded from above by $||P|| \times (69\nabla P - 1) + |P| \times 11$, as the number of positive body atoms in P is less than $||P||$ and the number of non-atomic rules is at most $|P|$. This makes $||\mathrm{Tr}_{\mathrm{MAX}}(P)||$ linear in $||P|| \times \nabla P$ and the time required to compute $\mathrm{Tr}_{\mathrm{MAX}}(P)$ is of order $||P|| \times \nabla P$, as $\mathrm{Tr}_{\mathrm{MAX}}(P)$ can be formed on a rule-by-rule basis.

4. The case of the translation $\mathrm{Tr}_{\mathrm{MIN}}(P)$ from Definition 6.11 is similar. A rule $r \in P$ is translated so that $||\mathrm{Tr}_{\mathrm{MIN}}(r, \nabla P)|| = 69\nabla P - 1$, too, and $||\mathrm{Tr}_{\mathrm{MIN}}(P)||$ is bounded from above by $|P| \times (69\nabla P - 1) + |\mathrm{Hb}(P)| \times 11$ which is also linear in $||P|| \times \nabla P$. The actual translation can be produced in time linear to $||P|| \times \nabla P$ by passing through P and $\mathrm{Hb}(P)$.

It remains to show the faithfulness of $\mathrm{Tr}_{\mathrm{AT}}$. Definition 6.12 implies that $\mathrm{Hb}(P) \subseteq \mathrm{Hb}(\mathrm{Tr}_{\mathrm{AT}}(P))$ and $\mathrm{Hb}_{\mathrm{v}}(\mathrm{Tr}_{\mathrm{AT}}(P)) = \mathrm{Hb}_{\mathrm{v}}(P)$. By Proposition 9.26 and Theorem 5.8, there is an extension function $\mathrm{Ext}_{\mathrm{AT}}$ that maps $M \in \mathrm{SM}(P)$ into $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#) \in \mathrm{SM}(\mathrm{Tr}_{\mathrm{AT}}(P))$ such that $M = N \cap \mathrm{Hb}(P)$. Recall that the level numbering $\#$ is uniquely determined in this relationship. Moreover, we know by Proposition 9.27 and Theorem 5.8 that if $N \in \mathrm{SM}(\mathrm{Tr}_{\mathrm{AT}}(P))$, then $M = N \cap \mathrm{Hb}(P) \in \mathrm{SM}(P)$ and $N = \mathrm{Ext}_{\mathrm{AT}}(P, M, \#)$ where $\#$ is the unique level numbering associated with M. Thus we may conclude $\mathrm{Tr}_{\mathrm{AT}}$ faithful by Theorem 3.19. □

HELSINKI UNIVERSITY OF TECHNOLOGY LABORATORY FOR THEORETICAL COMPUTER SCIENCE
RESEARCH REPORTS

HUT-TCS-A69 Marko Mäkelä
A Reachability Analyser for Algebraic System Nets. June 2001.

HUT-TCS-A70 Petteri Kaski
Isomorph-Free Exhaustive Generation of Combinatorial Designs. December 2001.

HUT-TCS-A71 Keijo Heljanko
Combining Symbolic and Partial Order Methods for Model Checking 1-Safe Petri Nets.
February 2002.

HUT-TCS-A72 Tommi Junttila
Symmetry Reduction Algorithms for Data Symmetries. May 2002.

HUT-TCS-A73 Toni Jussila
Bounded Model Checking for Verifying Concurrent Programs. August 2002.

HUT-TCS-A74 Sam Sandqvist
Aspects of Modelling and Simulation of Genetic Algorithms: A Formal Approach.
September 2002.

HUT-TCS-A75 Tommi Junttila
New Canonical Representative Marking Algorithms for Place/Transition-Nets. October 2002.

HUT-TCS-A76 Timo Latvala
On Model Checking Safety Properties. December 2002.

HUT-TCS-A77 Satu Virtanen
Properties of Nonuniform Random Graph Models. May 2003.

HUT-TCS-A78 Petteri Kaski
A Census of Steiner Triple Systems and Some Related Combinatorial Objects. June 2003.

HUT-TCS-A79 Heikki Tauriainen
Nested Emptiness Search for Generalized Büchi Automata. July 2003.

HUT-TCS-A80 Tommi Junttila
On the Symmetry Reduction Method for Petri Nets and Similar Formalisms.
September 2003.

HUT-TCS-A81 Marko Mäkelä
Efficient Computer-Aided Verification of Parallel and Distributed Software Systems.
November 2003.

HUT-TCS-A82 Tomi Janhunen
Translatability and Intranslatability Results for Certain Classes of Logic Programs.
November 2003.

ISBN 951-22-6832-9
ISSN 1457-7615