Journal of Artificial Intelligence Research 64 (2019) 749-815 Submitted 04/18; published 03/19

Polynomial and Exponential Bounded Logic Programs with Function Symbols: Some New Decidable Classes

Vernon Asuncion V.ASUNCION@WESTERNSYDNEY.EDU.AU
Yan Zhang YAN.ZHANG@WESTERNSYDNEY.EDU.AU
Artificial Intelligence Research Group
School of Computing, Engineering and Mathematics
Western Sydney University, Locked Bag 1797
Penrith, NSW 2751, Australia

Heng Zhang HENG.ZHANG@TJU.EDU.CN
College of Intelligence and Computing
Tianjin University, Tianjin 300050, China

Ruixuan Li RXLI@HUST.EDU.CN
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, Hubei 430074, China

Abstract

A logic program with function symbols is called finitely ground if there is a finite propositional logic program whose stable models are exactly the same as the stable models of this program. Finite groundability is an important property for logic programs with function symbols because it makes feasible to compute such programs' stable models using traditional ASP solvers. In this paper, we introduce new decidable classes of finitely ground programs called poly-bounded and k-EXP-bounded programs, which, to the best of our knowledge, strictly contain all other decidable classes of finitely ground programs discovered so far in the literature. We also study the relevant complexity properties for these classes of programs. We prove that the membership complexities for poly-bounded and k-EXP-bounded programs are EXPTIME-complete and (k+1)-EXPTIME-complete, respectively.

1. Introduction

A logic program Π with function symbols is called *finitely ground* if there is a finite propositional logic program Π' such that Π and Π' have exactly the same collection of stable models. Therefore, a finitely ground logic program will have a finite number of stable models and each stable model is of a finite size. Finite groundability is an important property for programs with function symbols because this makes feasible to compute such programs' stable models using traditional ASP solvers, as as done by Alviano, Faber and Leone (2010), Baselice, Bonatti and Criscuolo (2009), Calimeri, Cozza, Ianni and Leone (2008)

Unfortunately, in general, checking whether a program is finitely ground is undecidable. In recent years, several decidable classes of finitely ground programs have been discovered under the stable model semantics: *ω-restricted programs* (Syrjänen, 2001), *λ-restricted programs* (Gebser, Schaub, & Thiele, 2007), *finite domain programs* (Calimeri et al., 2008), *argument-restricted programs* (Lierler & Lifschitz, 2009), *safe programs* (Greco, Spezzano, & Trubitsyna, 2012), *Γ-acyclic programs* (Greco et al., 2012), and what we refer as GMT-

bounded programs, which has been shown to be a proper superclass of all other previous classes (Greco, Molinaro, & Trubitsyna, 2013). More recently, another decidable class of finitely ground programs called SIZE-*restricted programs* was further introduced by Calautti, Greco, Molinaro and Trubitsyna (2015a), where they showed that although this class does not properly contain the argument-restricted and GMT-bounded classes, the underlying technique may be combined with other approaches and eventually to identify more finitely ground programs.

In this paper, we further introduce two new decidable classes of logic programs called POLY-bounded and k-EXP-bounded, where POLY-bounded contains all the previous classes mentioned above and k-EXP-bounded contains POLY-bounded (see Figure 1 [1]).

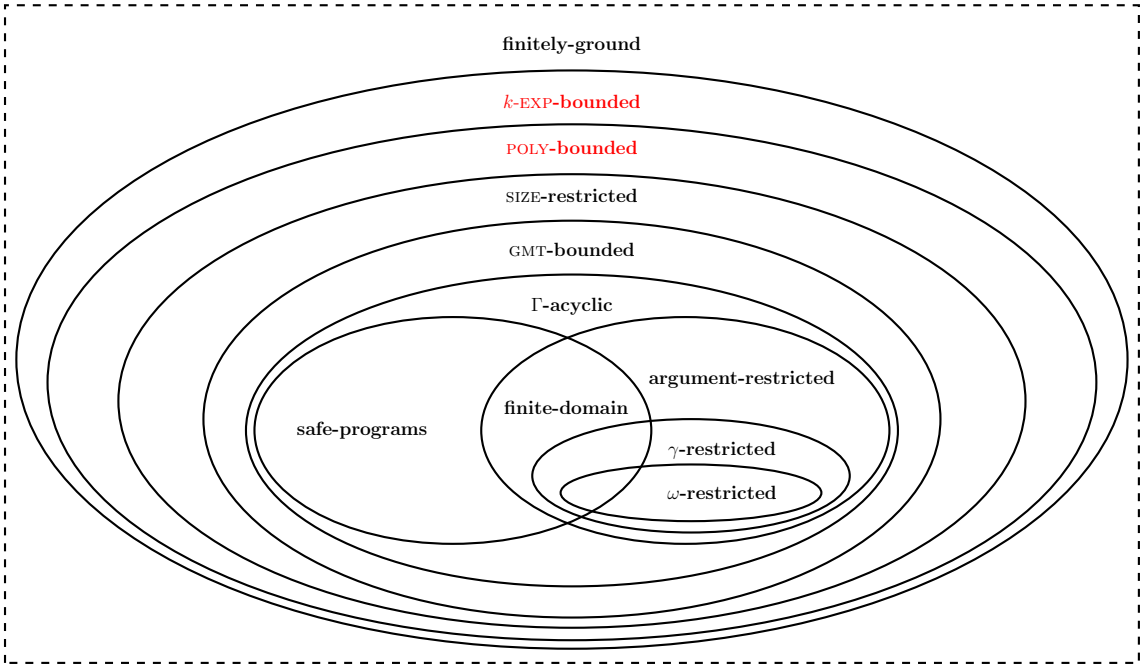

Figure 1: Landscape of decidable classes of logic programs with function symbols.

It has come to our attention that there are still some simple programs that are finitely ground but do not belong to either of the two classes GMT-bounded programs or SIZE-restricted programs. Let us consider a scenario of an online photo gallery, where each paid member can view any image in the gallery, but a restriction is imposed for guest members: Although a guest member is allowed to view the gallery images, he/she can only view no more than three images. Moreover, since only certain images are viewable, each such viewable image can only be succeeded through a unique image defined by the function "*next*". Also, in the case that the guest member skips a particular image, he/she cannot view the rest of the remaining images in the sequence (whose sequence is defined via the

1. As will be formally defined in Appendix A.2, we assume in Figure 1 that the notion of SIZE-restricted we use throughout the paper is the one combining with the GMT-bounded approach, which was shown in (Calautti et al., 2015a) to strictly contain the other classes discovered so far.

Decidable class	**ω-restricted**	**λ-restricted**	**finite-domain**	**argument-restricted**	**safe**
Membership complexity	PTIME	PTIME	PTIME	PTIME	PSPACE

Decidable class	**Γ-acyclic**	**gmt-bounded**	**size-restricted**	**poly-bounded**	**k-exp-bounded**
Membership complexity	PSPACE	PSPACE	PSPACE	EXPTIME-complete	$(k+1)$-EXPTIME-complete

Figure 2: Membership decision problem complexities.

function "*next*" mentioned above). This may be expressed by the following four rules:

$$\begin{aligned}
&\mathsf{r}_1: \ \mathsf{imageViewed}\big(\mathit{next}(X), Y\big) \leftarrow \mathsf{guestFirstViewed}(X, Y), \mathsf{guestMember}(Y),\\
&\mathsf{r}_2: \ \mathsf{skip}\big(\mathit{next}(X), Y\big) \vee \mathsf{imageViewed}\big(\mathit{next}(X), Y\big) \leftarrow \mathsf{imageViewed}(X, Y),\\
&\mathsf{r}_3: \ \bot \leftarrow \mathsf{imageViewed}\big(\mathit{next}\big(\mathit{next}\big(\mathit{next}\big(\mathit{next}(X)\big)\big)\big), Y\big).
\end{aligned}$$

Here: "$\mathsf{guestMember}(X)$" encodes that "X" is a guest; "$\mathsf{guestFirstViewed}(X, Y)$" encodes that guest "$Y$" *initially* viewed image "X"; "$\mathsf{imageViewed}(X, Y)$" encodes that guest "$Y$" viewed image "$X$"; "$\mathsf{skip}(X, Y)$" is a predicate that allows the guest member to skip the image "X"; and lastly, "$\mathit{next}(X)$", where "*next*" is a function symbol, encodes the function that returns the next image that should succeed "X" after having viewed "X", i.e., $\mathit{next}(X) = Y$ implies that after viewing "X", the next image can only be "Y".

Let $\Pi_1 = \{\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3\}$. Then it follows that the program Π_1 is finitely ground because program $\Pi_1 \cup D$ will only have finite stable models for any given input database D. Indeed, because of the constraint enforced by r_3, it follows that we cannot have more than three nesting of the function "*next*" within the atoms of the predicate "$\mathsf{imageViewed}$". That is, if for some term t, we let $\mathit{next}^k(t)$ denote the k-nesting of the function symbol *next*:

$$\underbrace{\mathit{next}(\mathit{next}(\mathit{next}(\ldots \mathit{next}(t)\ldots)))}_{k\text{-times}},$$

then we get that any atom of the form $\mathsf{imageViewed}\big(\mathit{next}^k(t_1), t_2\big)$ in any stable model of Π_1 must have $k < 4$ due to the constraint r_3. Because of this restriction, it follows that rule r_2 will only be limited in how it can recursively enforce the generation of complex terms through the head atom "$\mathsf{imageViewed}\big(\mathit{next}(X), Y\big)$" and the function symbol "*next*".

Surprisingly, it is not difficult to observe that Π_1 is not bounded according to the definition by Greco, Molinaro and Trubitsyna (2013), nor is it SIZE-restricted (Calautti et al., 2015a) because these (termination) criterions cannot detect that the atoms $\mathsf{imageViewed}\big(\mathit{next}^k(t_1), t_2\big)$ in a stable model of the program Π_1 must have $k < 4$. Indeed, the fact that the recursive rule "r_2" has the head atom "$\mathsf{imageViewed}\big(\mathit{next}(X), Y\big)$" with the complex term "$\mathit{next}(X)$", at that point probably not clear under their termination criteria.

Motivated by this example, this paper proposes yet another decidable class of logic programs with function symbols, called POLY-bounded programs, which strictly contains both

GMT-bounded and SIZE-restricted programs. A further generalization of the POLY-bounded programs leads to a new sequence of decidable classes called k-EXP-bounded programs for any integer $k \geq 0$. The reason that we are able to obtain such new classes is that by giving explicit treatments to "disjunctions", "negations" and "constraints" in the underlying programs, as well as leveraging the information about how atoms can possibly be derived from the program through a fixpoint-like termination criterion, we can derive proper upper bounds for the stable models of programs, and once such upper bounds satisfy certain restrictions, i.e., polynomial and exponential boundedness, respectively, it will lead to the underlying programs to become finitely ground. We further prove that the membership decision problem for the POLY-bounded class is EXPTIME-complete and is $(k+1)$-EXPTIME-complete for the k-EXP-bounded class (see Figure 2).

The rest of the paper is organized as follows: Section 2 presents necessary terminologies and background knowledge we will need throughout the paper. Section 3 defines fixpoint lower and upper bounds of all stable models for a given program with function symbols. Based on this upper bound definition, Section 4 then specifies a new decidable class of programs called polynomially bounded programs, and proves its main properties. Section 5 then generalizes the notion of polynomially bounded programs to a novel characterization of arbitrarily exponentially bounded programs called the k-EXP-bounded for any integer $k \geq 0$ such that the polynomially bounded class of programs corresponds exactly to the 0-EXP-bounded class. Section 6 studies the complexity property of these new decidable class of programs. Section 7 considers a case study where we show a particular use of exponentially bounded programs in the form of propositional planning under ASP. Finally, section 8 concludes the paper with some remarks.

2. Preliminaries

In this section, we introduce necessary concepts, notations and definitions that we will need throughout this paper. For a better flow of the development of the theories and techniques proposed in this paper, we refer the reader to Appendix A for the background notions about the GMT-bounded (Greco et al., 2013) and SIZE-restricted (Calautti et al., 2015a) classes of decidable logic programs with function symbols.

2.1 Symbols, Arguments, Terms, Atoms, Assignments and Grounding

We assume four *disjoint* infinite but *countable* sets of symbols $\mathcal{P}$, $\mathcal{V}$, Const and $\mathcal{F}$, standing for the sets of *predicates*, *variables*, *constants* and *functions* symbols, respectively. Each predicate symbol $p \in \mathcal{P}$ (function symbol $f \in \mathcal{F}$) is equipped with a natural number $n \in \mathbb{N}$ denoting its arity. For a predicate p (resp. function f) of arity n, the i-th *argument* of p (resp. f) is an expression of the form $p[i]$ (resp. $f[i]$)[2]. We further denote by $\text{ARITY}(p)$ $\big(\text{resp. } \text{ARITY}(f)\big)$ as p's (resp. f's) arity.

A *term* t is defined inductively as follows: (1) each constant $c \in \mathsf{Const}$ and variable $X \in \mathcal{V}$ are terms; (2) for each function symbol $f \in \mathcal{F}$ of arity n and any n-sequence of terms t_1, ..., t_n, we have that $f(t_1, \ldots, t_n)$ is also a term. Naturally, we say that a term t is *simple*

2. More formally, the i-th *argument* of p (f) is a pair (p, i) (resp. (f, i)) but written as "$p[i]$" (resp. "$f[i]$") for notational convenience.

if $t \in \mathcal{V} \cup \mathsf{Const}$ otherwise, we say that t is complex (i.e., t mentions some function symbols from $\mathcal{F}$ that may even be nested). Finally, an *atom* is a construct of the form $p(t_1, \ldots, t_n)$ where p is a predicate symbol from $\mathcal{P}$ and $\text{ARITY}(p) = n$, and each t_i (for $i \in \{1, \ldots, n\}$) is a term.

Given a term t, we denote by $\text{VAR}(t)$, $\text{CONST}(t)$ and $\text{FUNCT}(t)$ as the set of all the variables, constants and function symbols mentioned in t, respectively. We naturally extend this to an atom $A = p(t_1, \ldots, t_n)$ such that $\text{VAR}(A) = \bigcup_{i=1}^{i=n} \text{VAR}(t_i)$, $\text{CONST}(A) = \bigcup_{i=1}^{i=n} \text{CONST}(t_i)$ and $\text{FUNCT}(A) = \bigcup_{i=1}^{i=n} \text{FUNCT}(t_i)$. Additionally for convenience later on, we also denote by $\text{PRED}(A)$ as the predicate symbol of the atom A. Moreover, we also naturally extend these notions to a set of atoms S such that $\text{VAR}(S) = \bigcup_{A \in S} \text{VAR}(A)$, $\text{CONST}(S) = \bigcup_{A \in S} \text{CONST}(A)$ and $\text{FUNCT}(S) = \bigcup_{A \in S} \text{FUNCT}(A)$ denotes the set of variables, constants, atoms and function symbols occurring in S, respectively. For an atom $A = p(t_1, \ldots, t_n)$, we define $\text{TERMS}(A)$ as the union $\text{TERMS}(A) = \bigcup_{i=1}^{i=n} \text{terms}(t_i)$, and where for a term t, $\text{terms}(t)$ is defined inductively as follows: (1) if t is a constant c, then $\text{terms}(t) = c$; (2) if t is a variable X, then $\text{terms}(t) = X$; and (3) if t is a complex term $f(u_1, \ldots, u_m)$, then $\text{terms}(t) = \{f(u_1, \ldots, u_m)\} \cup \bigcup_{i=1}^{i=m} \text{terms}(u_i)$. Intuitively, for an atom A, $\text{TERMS}(A)$ is the set of all subterms mentioned in A. We also naturally extend this notion to a set S of atoms such that $\text{TERM}(S) = \bigcup_{A \in S} \text{TERMS}(A)$. For convenience, we further denote by $\text{VARCONST}(S)$ as the union $\text{VAR}(S) \cup \text{CONST}(S)$.

As usual, we say that an atom A (resp. set of atoms S) is *ground* if $\text{VAR}(A) = \emptyset$ (resp. $\text{VAR}(S) = \emptyset$), i.e., A (resp. S) does not mention any variables.

Now given our pairwise disjoint (infinite but countable) sets of variables $\mathcal{V}$, constants Const and function symbols $\mathcal{F}$ as mentioned above, let us further denote by $\mathcal{T}(\mathcal{V}, \mathsf{Const}, \mathcal{F})$ as the set of all the possible terms (and thus, is also an infinite set) that can be formed from the symbols $\mathcal{V}$, Const and $\mathcal{F}$ in accordance with our inductive "term" definition above. Then given a term $t \in \mathcal{T}(\mathcal{V}, \mathsf{Const}, \mathcal{F})$, an *assignment* θ is a function

$$\theta : \mathcal{T}(\mathcal{V}, \mathsf{Const}, \mathcal{F}) \longrightarrow \mathcal{T}(\mathcal{V}, \mathsf{Const}, \mathcal{F}),$$

where $\theta(t)$ is inductively defined as follows: (1) if t is a constant c, then $\theta(t) = c$, else; (2) if t is a variable X, then $\theta(t) = \theta(X)$, else; (3) if t is a complex term $f(t_1, \ldots, t_n)$, then $\theta(t) = f\big(\theta(t_1), \ldots, \theta(t_n)\big)$. Then naturally, we extend this notion to an atom $A = p(t_1, \ldots, t_n)$ such that $A\theta$ denotes the (transformed) atom $p\big(\theta(t_1), \ldots, \theta(t_n)\big)$. As folklore in the literatures, given two assignments $\theta_1 : S_1 \longrightarrow S_2$ and $\theta_2 : S_2 \longrightarrow S_3$, we denote by $(\theta_2 \circ \theta_1)$ as the composition of the two assignments θ_1 and θ_2 (which are also functions), respectively, such that $(\theta_2 \circ \theta_1) : S_1 \longrightarrow S_3$ and where $(\theta_2 \circ \theta_1)(x) = \theta_2\big(\theta_1(x)\big) \in S_3$, for each $x \in S_1$.

Given two atoms A_1 and A_2, we say that A_1 and A_2 are *unifiable* if there exist two assignments $(\theta_1 \circ \eta_1) : \text{VAR}(A_1) \longrightarrow \mathcal{T}(\mathcal{V}, \mathsf{Const}, \mathcal{F})$ and $(\theta_2 \circ \eta_2) : \text{VAR}(A_2) \longrightarrow \mathcal{T}(\mathcal{V}, \mathsf{Const}, \mathcal{F})$ where: (1) $\eta_1 : \text{VAR}(A_1) \longrightarrow \mathcal{V}$ and $\eta_2 : \text{VAR}(A_2) \longrightarrow \mathcal{V}$ are renaming (*bijective*) substitutions so that $\text{VAR}(A_1\eta_1) \cap \text{VAR}(A_2\eta_2) = \emptyset$; and (2) $\theta_1 : \text{VAR}(A_1\eta_1) \longrightarrow \mathcal{T}(\mathcal{V}, \mathsf{Const}, \mathcal{F})$ and $\theta_2 : \text{VAR}(A_2\eta_2) \longrightarrow \mathcal{T}(\mathcal{V}, \mathsf{Const}, \mathcal{F})$ such that $A_1(\theta_1 \circ \eta_1) = A_2(\theta_2 \circ \eta_2)$. Here, we note that we require the "initial" renaming substitutions η_1 and η_2 so that we do not create unwanted clashes of variable names between the two atoms A_1 and A_2. For example, assuming we have two atoms $A_1 = \mathsf{t}(f(X, X, Y))$ and $A_2 = \mathsf{t}(g(X, Y, Y))$, then we have that A_1 and A_2 are *not* unifiable. On the other hand, if we also have the two atoms $A_1' = \mathsf{t}(g(X, Y, Z, Y))$ and $A_2' = \mathsf{t}(V)$, then we have that A_1' and A_2' are unifiable, because if we let $\theta_1 = \{X \mapsto a,$

$Y \mapsto b$, $Z \mapsto c\}$ and $\theta_2 = \{V \mapsto g(a, b, c, b)\}$, where $\{a, b, c\} \subseteq \mathsf{Const}$, then we get that $A'_1\theta_1 = A'_2\theta_2$.

For a given set of atoms S and a set of constants C, we denote by $S\restriction_C$ as the ground set of atoms obtained from S by replacing each of the variables in S with constants from C in all possible ways. That is, we set $S\restriction_C = \{A\theta \mid \theta : \text{VAR}(A) \longrightarrow C\}$.

2.2 Terms Depth and Size

For a given term t, we denote by $\text{DEP}(t)$ as the *depth* of the term t defined inductively as follows

$$\text{DEP}(t) = \begin{cases} 0 & \text{if } t \text{ is a constant } c \text{ or a variable } X, \\ 1 + \text{MAX}\big(\{\text{DEP}(t_1), \ldots, \text{DEP}(t_n)\}\big) & \text{if } t = f(t_1, \ldots, t_n). \end{cases}$$

Intuitively, the depth of a term t corresponds to the maximum depth of nested functions mentioning variables within the term t, e.g., let $t_1 = f(f(X, g(c), Z), Y, Y)$, $t_2 = f(f(X, c, Z), Y, Y)$ and $t_3 = f(X, Y, Y)$, where $\{X, Y\} \subseteq \mathcal{V}$ are variables and $\{a, c\} \subseteq \mathsf{Const}$ are constants, then we have that $\text{DEP}(t_1) = 3$, $\text{DEP}(t_2) = 2$ and $\text{DEP}(t_3) = 1$. We naturally extend this to an atom $A = p(t_1, \ldots, t_n)$ and set of atoms S so that $\text{DEP}(A) = \text{MAX}\big(\{\text{DEP}(t_1) \ldots \text{DEP}(t_n)\}\big)$ and $\text{DEP}(S) = \text{MAX}\big(\{\text{DEP}(A) \mid A \in S\}\big)$, respectively.

Also, for convenience later in our study in section 4, we define the *ground-size* of the atom A, where $A = p(t_1, \cdots, t_n)$, denoted as $\text{GRSIZE}(A)$, to be $\text{GRSIZE}(A) = \Sigma_{i=1}^{n}\text{grsize}(t_i)$, where for term t, $gr\text{SIZE}(t)$ is defined inductively as follows:

$$\text{grsize}(t) = \begin{cases} 0 & \text{if } t \text{ is a constant } c \text{ or a variable } X, \\ m + \sum_{i=1}^{m} \text{grsize}(t_i) & \text{if } t = f(t_1, \ldots, t_m). \end{cases}$$

2.3 Disjunctive Logic Programs, Range Restrictions, Input Databases and Intensional/Extensional Predicates

A *disjunctive logic program* (or simply called *program*) Π is a finite set of *rules* r of the form:

$$A_1 \vee \ldots \vee A_k \leftarrow B_1, \ldots, B_l, \mathsf{not}\, C_1, \ldots, \mathsf{not}\, C_m, \tag{1}$$

where A_i, B_j, C_h are atoms for all $1 \leq i \leq k, 1 \leq j \leq l$ and $1 \leq h \leq m$. We denote by $Hd(r)$, $Pos(r)$, and $Neg(r)$ the sets $\{A_1, \ldots, A_k\}$, $\{B_1, \ldots, B_l\}$, and $\{C_1, \ldots, C_m\}$, which are called r's *head*, *positive body*, and *negative body*, respectively. Sometimes for convenience, we may simply denote rule r by $Hd(r) \leftarrow Pos(r) \wedge \neg Neg(r)$. When $k \leq 1$ for all $r \in \Pi$, Π is called a *normal program*; if also $k = 1$ and $Neg(r) = \emptyset$ for all $r \in \Pi$, Π is called a *positive normal program*; if for a rule r, $k = 0$, r is called a *constraint* and we denote its head by $\bot$; and when $Pos(r) \cup Neg(r) = \emptyset$ and $|Hd(r)| = 1$, then r is called a *fact*.

We denote by $\text{ARG}(\Pi)$ as the set of all arguments[3] of all predicates occurring in Π, $\text{PRED}(\Pi)$ as the set of all predicate symbols in Π, and $\text{ATOMS}(\Pi)$ as the set of all atoms mentioned in Π. We further denote by $\text{HD}(\Pi) = \bigcup_{r\in\Pi} Hd(r)$ and $\text{POS}(\Pi) = \bigcup_{r\in\Pi} Pos(r)$

3. Recall an argument, defined in the beginning of section 2.1, is a position of a predicate, denoted as $p[i]$, where p is a predicate, and i is the position of the i-th term occurring in p.

as the union of the head atoms and positive body atoms mentioned in Π, respectively. Similarly as defined in section 2.1 for the case of atoms and sets of atoms, we also denote by $\text{VAR}(\Pi)$, $\text{CONST}(\Pi)$ and $\text{FUNCT}(\Pi)$ as the sets of variables, constants and function symbols occurring in the program Π, respectively. Similarly, we also denote by $\text{VARCONST}(\Pi)$ as the union $\text{VAR}(\Pi) \cup \text{CONST}(\Pi)$. A program Π is *ground* if $\text{VAR}(\Pi) = \emptyset$, i.e., Π does not mention any variables.

A program Π is *range restricted* if for each rule, the variables occurring in the head or in the negative body also appear in the positive body of that rule. As showed by Greco, Molinaro and Trubitsyna (2013), in this paper, we assume that all programs are range restricted. For a given program Π, a finite set D of facts (D can be empty) is called an *input database* of Π when we consider program $\Pi \cup D$.

We assume that predicate symbols in $\text{PRED}(\Pi)$ are partitioned into two different classes as follows: (1) *extensional* (or *input*) predicate symbols, whose relations can only be derived from facts; and (2) *intensional* predicates symbols, whose relations can only be derived from rules with non-empty bodies. For convenience, we denote by $\text{EXT}(\Pi)$ and $\text{INT}(\Pi)$, as the set of those extensional and intensional predicate symbols previously mentioned, respectively.

2.4 Stable Models and Limited Arguments of Programs

Now for a given program Π, by $\text{HU}(\Pi)$ and $\text{HB}(\Pi)$, we denote Π's *Herbrand universe* and *Herbrand base*, respectively. Specifically, $\text{HU}(\Pi)$ is the set of all ground terms that are built using the constants and function symbols in Π (if Π does not contain any constant, we introduce a constant in Π's domain), while $\text{HB}(\Pi)$ is the set of all atoms that care built from terms in $\text{HU}(\Pi)$ and predicate symbols of Π. Clearly, both $\text{HU}(\Pi)$ and $\text{HB}(\Pi)$ can be infinite. We say that a set I of atoms is an *interpretation* of Π iff $I \subseteq \text{HB}(\Pi)$. A rule r' is a *ground instance* of $r \in \Pi$ if r' is obtained from r by substituting each variable in r by some ground term from $\text{HU}(\Pi)$. We use $\text{GROUND}(r)$ to denote all ground instances of r, and $\text{GROUND}(\Pi) = \bigcup_{r \in \Pi} \text{GROUND}(r)$ as the *grounding* of the program Π, which could be infinite. Given an interpretation I and a ground rule r', we say that I *satisfies* r', denoted as $I \models r'$, iff $I \cap Hd(r') \neq \emptyset$ whenever both $I \subseteq Pos(r')$ and $I \cap Neg(r') = \emptyset$ holds. Then we say that I is a *model* of a ground program Π', denoted as $I \models \Pi'$, iff $I \models r'$, for all $r' \in \Pi'$.

Given an interpretation $I \subseteq \text{HB}(\Pi)$ and the grounding $\Pi' = \text{GROUND}(\Pi)$ of Π, we denote by $(\Pi')^I$ as the *reduced* (or *reduct*) of the (ground) program Π' with respect to I such that it is denoted as the set of rules $\{Hd(r) \leftarrow Pos(r) \mid r \in \Pi' \text{ and } Neg(r) \cap I = \emptyset\}$. Then we say that I is a *stable model* of Π iff I is the minimal set that satisfies all the rules in $(\Pi')^I$ (Gelfond & Lifschitz, 1988, 1991).

An argument $p[i]$ in $\text{ARG}(\Pi)$ is said to be *limited* iff for every finite set of facts D and for every stable model M of program $\Pi \cup D$, the set $\{t_i \mid p(t_1, \cdots, t_i, \cdots, t_n) \in M\}$ is finite. Moreover, program Π is said to be *limited* iff every argument in $\text{ARG}(\Pi)$ is limited.

2.5 Embeddings of Ground Atoms

Given a set of *ground* atoms S_1, i.e., $\text{VAR}(S_1) = \emptyset$) and a set of atoms (not necessarily ground) S_2, we say that S_1 *is embeddable* into S_2, denoted as $S_1 \ll S_2$, iff for each atom

$A_1 \in S_1$, there exist an atom $A_2 \in S_2$ and an assignment $\theta : \text{VAR}(A_2) \longrightarrow \text{TERMS}(A_1)$ such that $A_2\theta = A_1$. Naturally, we denote by $S_1 \not\ll S_2$ if S_1 is *not* embeddable into S_2.

Intuitively, if $S_1 \ll S_2$, then each ground atom contained in S_1 will have a corresponding (not necessarily ground) atom in S_2. In other words, S_2 represents no less information than S_1 does. For instance, let $S_1 = \{p\big(a, f(f(b)), f(f(b))\big),\ q(b,b,c),\ r(a),\ r(b),\ r(c)\}$ be a set of ground atoms, and $S_2 = \{p\big(X, f(f(Y)), f(f(Y))\big),\ q(V,W,X),\ r(W)\}$ and $S_2' = \{p\big(X, f(f(X)), f(f(Y))\big),\ q(V,V,X),\ r(U)\}$ be two sets of atoms. Then we have that $S_1 \ll S_2$ but $S_1 \not\ll S_2'$.

It is observed that $S_1 \ll S_2$, because for each of the ground atoms $p\big(a, f(f(b)), f(f(b))\big)$, $q(b,b,c)$, $r(a)$, $r(b)$, $r(c) \in S_1$ and atoms $p\big(X, f(f(Y)), f(f(Y))\big)$, $q(V,W,X)$, $r(W) \in S_2$, we can have that $p\big(a, f(f(b)), f(f(b))\big) = p\big(X, f(f(Y)), f(f(Y))\big)\theta_{21}$, $q(b,b,c) = q(V,W,X)\theta_{22}$, $r(a) = r(W)\theta_{23}$, $r(b) = r(W)\theta_{24}$ and $r(c) = r(W)\theta_{25}$, where the assignments $\theta_{21}, \ldots. \theta_{25}$ are defined as: $\theta_{21} = \{X \mapsto a,\ Y \mapsto b\}$, $\theta_{22} = \{V \mapsto b,\ W \mapsto b,\ X \mapsto c\}$, $\theta_{23} = \{W \mapsto a\}$, $\theta_{24} = \{W \mapsto b\}$ and $\theta_{25} = \{W \mapsto c\}$.

On the other hand, we note that $S_1 \not\ll S_2'$, because for the atom $A_1 = p\big(a, f(f(b)), f(f(b)\big)\big) \in S_1$, we cannot find any atom $A_2 \in S_2'$ and assignment $\theta : \text{VAR}(A_2) \longrightarrow \text{TERMS}(A_1)$ for which $A_1 = A_2\theta$. More precisely, since the only candidate of such atom A_2 from S_2' is $p\big(X, f(f(X)), f(f(Y))\big)$, then it is not too difficult to see that we cannot have such an assignment θ from the variables X, Y and Z of the atom A_2 onto the constants a and b of the atom A_1, because A_1 and A_2 are not unifiable. Note though that if we have the atom $A_2 = p\big(X, V, f(f(Y))\big)$ instead of $p\big(X, f(f(X)), f(f(Y))\big)$, then by setting the assignment θ as $\{X \mapsto a,\ V \mapsto f(f(b)),\ Y \mapsto b\}$, we have that $A_1 = A_2\theta$.

2.6 Definite Program Normal Form, Firing Graphs, Strongly Connected Components (SCCs) and Argumentation Graphs

For a given program Π, we define its *definite normal form*, denoted as Π^{DEF}, to be the program obtained from Π via the following transformation:

$$\big\{Hd(r') \leftarrow Pos(r') \wedge \neg Neg(r') \mid r' \text{ is a rule such that } \exists r \in \Pi \text{ where:}$$
$$(1)\ Hd(r') \in Hd(r);$$
$$(2)\ Pos(r') = Pos(r);$$
$$(3)\ Neg(r') = Neg(r) \cup \big(Hd(r) \setminus \{Hd(r')\}\big)\big\}.$$

Intuitively, Π^{DEF} (with "DEF" standing for *definite program*) is the normal program obtained from Π by "shifting" (Dix, Gottlob, & Marek, 1996). That is, for each rule r' of Π^{DEF}, only one atom still occurs in the head of r', while all other head atoms in the corresponding rule r in Π are shifted to the negative body of the new generated rule r'.

Given a program Π and its definite normal form Π^{DEF}, we denote by $\Omega(\Pi^{\text{DEF}})$ as the *firing graph* of Π^{DEF}, which is a *directed* graph such that with $\Omega(\Pi^{\text{DEF}}) = (V, E)$, $V = \Pi^{\text{DEF}}$ (i.e., the vertices of $\Omega(\Pi^{\text{DEF}})$ are the rules of Π^{DEF} itself) and there is an edge $\langle r, r'\rangle \in E$ iff the head atom $Hd(r)$ and some body atom in $Pos(r')$ are unifiable. Intuitively, this means that r may cause r' to "fire."

We say that a rule $r \in V$ (where $V = \Pi$) is in a *cycle* if there is a path in the firing graph $\Omega(\Pi^{\text{DEF}})$ such that r is reachable to itself. Then a *strongly connected component (SCC)* of

$\Omega(\Pi^{\text{DEF}})$ is a *maximal set* of nodes $\mathcal{C} \subseteq \Pi^{\text{DEF}}$ where every node of $\mathcal{C}$ is in a cycle. In this case, we say that such a set of nodes $\mathcal{C}$ is an SCC of the program Π^{DEF}. Note that since each individual node trivially reaches itself, even a single node itself can be an SCC, in this case we will call it a *trivial* SCC. For convenience, we denote by $\text{SCC}(\Pi^{\text{DEF}})$ as the set $\{\mathcal{C}_1, \ldots, \mathcal{C}_s\}$, which contains exactly the SCCs $\mathcal{C}_1, \ldots, \mathcal{C}_s$ of $\Omega(\Pi^{\text{DEF}})$. For two distinct SCCs $\mathcal{C}$, $\mathcal{C}' \in \text{SCC}(\Pi^{\text{DEF}})$ (i.e., $\mathcal{C} \neq \mathcal{C}'$), we say that $\mathcal{C}'$ *depends* on $\mathcal{C}$, denoted $\mathcal{C} \prec \mathcal{C}'$, if there exists some rules $r \in \mathcal{C}$ and $r' \in \mathcal{C}'$ such that $Hd(r)$ and some atom in $Pos(r')$ are *unifiable*.

The *argumentation graph* of the program Π^{DEF}, where Π^{DEF} is the definite normal form of Π), denoted as $\Delta(\Pi^{\text{DEF}}) = (V, E)$, is a directed graph with nodes $V = \text{ARG}(\Pi^{\text{DEF}})$, and there is an edge $(q[j], p[i]) \in E$ iff there exists some rule $r \in \Pi^{\text{DEF}}$, such that

1. there exists some atom $p(t_1, \ldots, t_i, \ldots, t_n)$ such that $p(t_1, \ldots, t_i, \ldots, t_n) = Hd(r)$;

2. there exists some positive body atom $q(u_1, \ldots, u_j, \ldots, u_m) \in Pos(r)$,

and the terms t_i and u_j share a common variable. Intuitively, an edge $(q[j], p[i])$ of $\Delta(\Pi^{\text{DEF}})$ indicates that there can be a derivation of terms from argument $q[j]$ to argument $p[i]$. We further say that an argument $p[i]$ *depends on* an argument $q[j]$ if there is a path from $q[j]$ to $p[i]$ in the graph $\Delta(\Pi^{\text{DEF}})$.

2.7 Recursive Rules/Atoms and SCC Stratification

For a given program Π, Π^{DEF} its definite normal form and some rule $r \in \Pi^{\text{DEF}}$, we say that the head atom $Hd(r)$ is *mutually recursive* with a body atom $B \in Pos(r)$ if there exists an SCC $\mathcal{C}$ of the firing graph $\Omega(\Pi^{\text{DEF}})$ which contains r, and there exists another rule r' also in $\mathcal{C}$ (which can also be r) such that $Hd(r')$ and B are unifiable. We denote by $recrPos(r)$ as the set of mutually recursive atoms in $Pos(r)$ with $Hd(r)$.

Symmetrically, we denote by $domPos(r)$ the subset of the (remaining) positive body atoms $Pos(r) \backslash recrPos(r)$ (i.e., $domPos(r) \subseteq Pos(r) \backslash recrPos(r)$) such that $B \in domPos(r)$ implies $\text{PRED}(B) \in \text{INT}(\Pi)$. Here, we note that if r is in some SCC $\mathcal{C}$ of the firing graph $\Omega(\Pi^{\text{DEF}})$, then the atoms of $domPos(r)$ are only derivable from the heads of rules belonging to some other SCC $\mathcal{C}'$ that is not $\mathcal{C}$, i.e., $\mathcal{C}' \neq \mathcal{C}$. Note that it is possible that the union $domPos(r) \cup recrPos(r)$ may not necessarily mention all the atoms of $Pos(r)$. In such a case, as we assumed in Section 2.3, we view those remaining atoms as of the extensional predicate symbols $\text{EXT}(\Pi)$ of Π, i.e., $\text{PRED}\Big(Pos(r) \Big\backslash \big(domPos(r) \cup recrPos(r)\big)\Big) \subseteq \text{EXT}(\Pi)$. As usual in the literatures (e.g., Greco et al., 2013), these atoms of the extensional relations are derived via the so-called *database facts*. For convenience, we denote by $extPos(r)$ as the set of atoms in $Pos(r) \Big\backslash \big(domPos(r) \cup recrPos(r)\big)$ of a rule r.

Let Π be a program, Π^{DEF} its definite normal form, $\Omega(\Pi^{\text{DEF}})$ the firing graph of Π^{DEF} and $\text{SCC}(\Pi^{\text{DEF}})$ the set of all SCCs of $\Omega(\Pi^{\text{DEF}})$. Then an *SCC stratification* of $\text{SCC}(\Pi^{\text{DEF}})$ is a sequence $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$, ..., $\text{SCC}(\Pi^{\text{DEF}})^{[K]}$ such that each of the $\text{SCC}(\Pi^{\text{DEF}})^{[i]}$ (for $i \in \{0, \ldots, K\}$) are subsets of $\text{SCC}(\Pi^{\text{DEF}})$ $\big($i.e., $\text{SCC}(\Pi^{\text{DEF}})^{[i]} \subseteq \text{SCC}(\Pi^{\text{DEF}})\big)$, and are defined

inductively as follows:

$$\begin{aligned}
\textsc{scc}(\Pi^{\textsc{def}})^{[0]} = \big\{\mathcal{C} \mid\ & \mathcal{C} \in \textsc{scc}(\Pi^{\textsc{def}}) \text{ and there does not exists some rule } r \in \mathcal{C} \text{ and} \\
& \text{atom } B \in Pos(r) \text{ such that for (some other) } \mathcal{C}' \in \textsc{scc}(\Pi^{\textsc{def}}) \big\backslash \{\mathcal{C}\} \\
& \text{and rule } r' \in \mathcal{C}', \text{ and } Hd(r') \text{ and } B \text{ are unifiable} \big\}; \qquad (2)
\end{aligned}$$

$$\begin{aligned}
\textsc{scc}(\Pi^{\textsc{def}})^{[i+1]} = \Big\{\mathcal{C} \mid\ & \mathcal{C} \in \textsc{scc}(\Pi^{\textsc{def}}) \Big\backslash \Big(\bigcup_{j=0}^{j=i} \textsc{scc}(\Pi^{\textsc{def}})^{[j]} \Big) \text{ and there exists some rule } r \in \mathcal{C} \\
& \text{and atom } B \in Pos(r) \text{ such that for some } \mathcal{C}' \in \textsc{scc}(\Pi^{\textsc{def}})^{[i]} \\
& \text{and rule } r' \in \mathcal{C}', \text{ and } Hd(r') \text{ and } B \text{ are unifiable} \Big\} \qquad (3)
\end{aligned}$$

(for each $i \in \{0, \ldots, K-1\}$).

Basically, the SCC stratification of $\textsc{scc}(\Pi^{\textsc{def}})$ groups and orders the elements of $\textsc{scc}(\Pi^{\textsc{def}})$ (i.e., the SCCs) into the layers corresponding to the sequence $\textsc{scc}(\Pi^{\textsc{def}})^{[0]}$, …, $\textsc{scc}(\Pi^{\textsc{def}})^{[K]}$, such that for each $i \in \{0, \ldots, K-1\}$, the set of SCCs $\textsc{scc}(\Pi^{\textsc{def}})^{[i+1]}$ contains exactly those SCC $\mathcal{C}$ that depends on some SCC $\mathcal{C}' \in \textsc{scc}(\Pi^{\textsc{def}})^{[i]}$ of the previous layer. Obviously, $\textsc{scc}(\Pi^{\textsc{def}})$ is finite because Π is finite. Also note that $\textsc{scc}(\Pi^{\textsc{def}})^{[0]}$, $\cdots$, $(\Pi^{\textsc{def}})^{[K]}$ form a partition of the set $\textsc{scc}(\Pi^{\textsc{def}})$.

We note that the union of the SCC stratification $\textsc{scc}(\Pi^{\textsc{def}})^{[0]} \cup \ldots \cup \textsc{scc}(\Pi^{\textsc{def}})^{[K]} = \bigcup_{i=0}^{i=K} \textsc{scc}(\Pi^{\textsc{def}})^{[i]}$ may not necessarily contain all the SCCs of $\textsc{scc}(\Pi^{\textsc{def}})$. This is because it is possible that some $\mathcal{C} \in \textsc{scc}(\Pi^{\textsc{def}})$ does not have any rule r in it that has some intensional body atom B that unifies with some head atom $Hd(r')$ of a rule $r' \in \Pi^{\textsc{def}}$. We note in this case that all the rules in $\mathcal{C}$ are "dead" or *redundant rules*.

For instance, if we assume $\Pi = \big\{\mathsf{r}_1 :\ \mathsf{p}(h(X)) \leftarrow \mathsf{q}(X),\ \mathsf{r}_2 :\ \mathsf{p}\big(g\big(g(X)\big)\big) \leftarrow \mathsf{p}(g(X))\big\}$ (note that Π is already in definite normal form and $\mathsf{q} \in \textsc{ext}(\Pi)$ is an extensional predicate symbol), then $\textsc{scc}(\Pi^{\textsc{def}}) = \big\{\{\mathsf{r}_1\}, \{\mathsf{r}_2\}\big\}$. Also note that SCC $\{\mathsf{r}_2\}$ does not depend on SCC $\{\mathsf{r}_1\}$ because $Hd(\mathsf{r}_1) = \mathsf{p}(h(X))$ is not unifiable with the only body atom $\mathsf{p}(g(X)) \in Pos(\mathsf{r}_2)$. In this case, it is easy to see that rule r_2 will not play any part in the derivation of "well-founded" atoms and thus, is a dead rule.

For convenience, when it is clear from the context, we will simply denote a rule r in some SCC $\mathcal{C} \in \textsc{scc}(\Pi^{\textsc{def}})^{[i]}$ of the stratum $\textsc{scc}(\Pi^{\textsc{def}})^{[i]}$ (for $i \in \{1, \ldots, K\}$) of the SCC stratification: $\textsc{scc}(\Pi^{\textsc{def}})^{[0]}$, …, $\textsc{scc}(\Pi^{\textsc{def}})^{[K]}$, by the notion $r \in \textsc{scc}(\Pi^{\textsc{def}})^{[i]}$, instead of stating that for some $\mathcal{C} \in \textsc{scc}(\Pi^{\textsc{def}})^{[i]}$, $r \in \mathcal{C}$.

Example 1. Assume our program Π to be the set consisting of the following rules:

$$\mathsf{r}_1 :\ \mathsf{p}(X) \leftarrow \mathsf{u}(X), \qquad (4)$$
$$\mathsf{r}_2 :\ \mathsf{q}(f(X)) \leftarrow \mathsf{q}(X), \mathsf{p}(X), \mathsf{t}(X), \qquad (5)$$
$$\mathsf{r}_3 :\ \mathsf{r}(f(X)) \leftarrow \mathsf{r}(X), \mathsf{q}(X), \mathsf{t}(X), \qquad (6)$$
$$\mathsf{r}_4 :\ \mathsf{s}(f(X)) \leftarrow \mathsf{s}(X), \mathsf{r}(X), \mathsf{t}(X). \qquad (7)$$

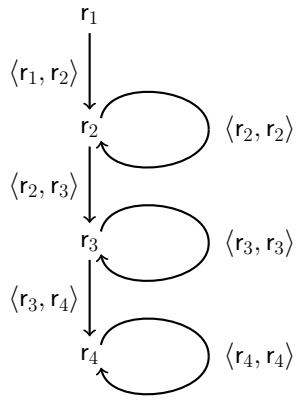

Figure 3: The graph $\Omega(\Pi^{\text{DEF}})$ of the rules r_1-r_4 from Example 1.

Note that $\Pi^{\text{DEF}} = \Pi$), and t and u are the only extensional predicate symbols. Consider $\Omega(\Pi^{\text{DEF}}) = (V, E)$, where $V = \{\mathsf{r}_1,\ \mathsf{r}_2,\ \mathsf{r}_3,\ \mathsf{r}_4\}$, and $E = \big\{\langle \mathsf{r}_1, \mathsf{r}_2\rangle,\ \langle \mathsf{r}_2, \mathsf{r}_2\rangle,\ \langle \mathsf{r}_2, \mathsf{r}_3\rangle,\ \langle \mathsf{r}_3, \mathsf{r}_3\rangle,\ \langle \mathsf{r}_3, \mathsf{r}_4\rangle,\ \langle \mathsf{r}_4, \mathsf{r}_4\rangle\big\}$ (see Figure 3).

It is further noted that $\text{SCC}(\Pi^{\text{DEF}}) = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\}$, where $\mathcal{C}_1 = \{\mathsf{r}_1\}$, $\mathcal{C}_2 = \{\mathsf{r}_2\}$, $\mathcal{C}_3 = \{\mathsf{r}_3\}$, and $\mathcal{C}_4 = \{\mathsf{r}_4\}$. Then it follows that for each rule r_i $(i = 1, 2, 3, 4)$, we have

(1) $domPos(\mathsf{r}_1) = \emptyset$, $recrPos(\mathsf{r}_1) = \emptyset$ and $extPos(\mathsf{r}_1) = \{\mathsf{u}(X)\}$;
(2) $domPos(\mathsf{r}_2) = \{\mathsf{p}(X)\}$, $recrPos(\mathsf{r}_2) = \{\mathsf{q}(X)\}$ and $extPos(\mathsf{r}_2) = \{\mathsf{t}(X)\}$;
(3) $domPos(\mathsf{r}_3) = \{\mathsf{q}(X)\}$, $recrPos(\mathsf{r}_3) = \{\mathsf{r}(X)\}$ and $extPos(\mathsf{r}_3) = \{\mathsf{t}(X)\}$; and
(4) $domPos(\mathsf{r}_4) = \{\mathsf{r}(X)\}$, $recrPos(\mathsf{r}_4) = \{\mathsf{s}(X)\}$ and $extPos(\mathsf{r}_4) = \{\mathsf{t}(X)\}$.

Finally, by setting $\text{SCC}(\Pi^{\text{DEF}})^{[0]} = \{\mathcal{C}_1\}$, $\text{SCC}(\Pi^{\text{DEF}})^{[1]} = \{\mathcal{C}_2\}$, $\text{SCC}(\Pi^{\text{DEF}})^{[2]} = \{\mathcal{C}_3\}$, and $\text{SCC}(\Pi^{\text{DEF}})^{[3]} = \{\mathcal{C}_4\}$, respectively, we obtain the sequence of sets $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$, $\text{SCC}(\Pi^{\text{DEF}})^{[1]}$, $\text{SCC}(\Pi^{\text{DEF}})^{[2]}$, $\text{SCC}(\Pi^{\text{DEF}})^{[3]}$, which forms an SCC stratification of $\text{SCC}(\Pi^{\text{DEF}})$. □

2.8 Restricted Arguments

We will need the notion of restricted arguments introduced in previous works (Greco et al., 2013; Lierler & Lifschitz, 2009).

Definition 1. *The set of* restricted arguments*, denoted as* $\text{AR}(\Pi)$*, is the set of arguments of* Π*, i.e.,* $\text{AR}(\Pi) \subseteq \text{ARG}(\Pi)$*, such that there exists an assignment* $\phi : \text{ARG}(\Pi) \longrightarrow \mathbb{N}$*, where for each rule* $r \in \Pi$*, each atom* $p(t_1, \ldots, t_n) \in Hd(r)$*, and each variable* X *of* t_i*,* $p[i] \in \text{AR}(\Pi)$ *iff*

(1) there is $q(u_1, \ldots, u_m) \in Pos(r)$ *with* X *occurring in* u_j*, and*

(2) $\phi\big(p[i]\big) - \phi\big(q[j]\big) \geq \text{DEP}\big(X, t_i\big) - \text{DEP}\big(X, u_j\big)$.

Here $\text{DEP}\big(X, t_i\big)$ $\big($*resp.* $\text{DEP}\big(X, u_j\big)\big)$ *denotes the maximum term depth of the variable* X *in term* t_i *(resp.* u_j*).*

Generally speaking, the restricted arguments from $\text{AR}(\Pi)$ of Π are the arguments where their edges in the argumentation graph do not induce a recursive "growing cycle" of complex terms.

Example 2. Assume Π and Π' to be two programs such that Π contains the single rule:

$$\mathsf{r}: \ \mathsf{p}\big(f(X)\big) \leftarrow \mathsf{p}(X),$$

and Π' the rules:

$$\mathsf{r}'_1: \ \mathsf{p}\big(f(Y)\big) \leftarrow \mathsf{q}(Y),$$
$$\mathsf{r}'_2: \ \mathsf{q}(Z) \leftarrow \mathsf{p}\big(f(Z)\big).$$

Then we have that $\text{AR}(\Pi) = \emptyset$ because for any assignment $\phi : \big\{\mathsf{p}[1]\big\} \longrightarrow \mathbb{N}$, it is impossible to satisfy the inequality:

$$\phi\big(\mathsf{p}[1]\big) - \phi\big(\mathsf{p}[1]\big) \geq \text{DEP}\big(X, f(X)\big) - \text{DEP}\big(X, X\big) \tag{8}$$

because, since $\phi\big(\mathsf{p}[1]\big) - \phi\big(\mathsf{p}[1]\big) = 0$ and $\text{DEP}\big(X, f(X)\big) - \text{DEP}\big(X, X\big) = 1 - 0 = 1$, for the terms "$f(X)$" and "$X$" in the head "$\mathsf{p}\big(f(X)\big)$" and body "$\mathsf{p}(X)$" of r', respectively, then we have that (8) becomes $0 \geq 1$, which is clearly false.

On the other hand, we have that $\text{AR}(\Pi') = \big\{\mathsf{p}[1], \mathsf{q}[1]\big\} = \text{ARG}(\Pi')$ because, if we choose the assignment $\phi : \big\{\mathsf{p}[1], \mathsf{q}[1]\big\} \longrightarrow \mathbb{N}$ such that $\phi\big(\mathsf{p}[1]\big) = 1$ and $\phi\big(\mathsf{q}[1]\big) = 0$, then we have that the two inequalities:

$$\phi\big(\mathsf{p}[1]\big) - \phi\big(\mathsf{q}[1]\big) \geq \text{DEP}\big(Y, f(Y)\big) - \text{DEP}\big(Y, Y\big), \tag{9}$$
$$\phi\big(\mathsf{q}[1]\big) - \phi\big(\mathsf{p}[1]\big) \geq \text{DEP}\big(Z, Z\big) - \text{DEP}\big(Z, f(Z)\big), \tag{10}$$

corresponding to the head variables "Y" and "Z" of the rules r'_1 and r'_2, respectively, becomes:

$$1 - 0 \geq 1 - 0 \ \equiv \ 1 \geq 1, \tag{11}$$
$$0 - 1 \geq 0 - 1 \ \equiv \ -1 \geq -1. \tag{12}$$

respectively, which is clearly true. □

3. Lower and Upper Bounds for Progression

Our idea of discovering a new decidable class of programs is described as follows: for a given program Π: (1) we first specify a lower bound $\mathcal{L}(\Pi)$ of the progression corresponding to all facts that definitely will be derived from Π; and (2) then using the lower bound $\mathcal{L}(\Pi)$, we further propose another progression based procedure to specify an approximation upper bound $\mathcal{U}(\Pi)$ for all stable models S of program $\Pi \cup D$ for all input databases D (D can be empty); and finally (3) by imposing a proper *polynomial* bound on $\mathcal{U}(\Pi)$, we are eventually able to derive a new decidable class of programs with function symbols that are finitely ground.

Now for a given program Π and its definite normal form Π^{DEF} as defined in Section 2.6, and via the Herbrand base of Π, we specify a procedure that computes the set of all facts that must be true or false derived from Π, i.e., the "lower bound" of the stable models of Π.

3.1 Deriving Lower Bound

In this section, we first propose a notion of how we derive such a "lower bound" as described above.

Definition 2. ***[Deriving lower bound]*** *Let* Π *be a program and* Π^{DEF} *be its definite normal form as described in Section 2.7. Then* $\mathcal{L}^k(\Pi)$ *(*$k \geq 0$*) is inductively defined as follows*[4]*:*

$$\begin{aligned}\mathcal{L}^0(\Pi) = &\big\{\langle\top,+\rangle,\ \langle\bot,-\rangle\big\} \cup \qquad (13)\\ &\big\{\ \langle A,+\rangle \ \big|\ \textit{there exists a rule ``}A \leftarrow \top\textit{'' } \in \Pi^{\text{DEF}} \textit{ and } A \textit{ is a ground atom}\ \big\} \cup\\ &\big\{\ \langle A\theta,-\rangle \ \big|\ \textit{there exist a constraint ``}\bot \leftarrow A\textit{'' } \in \Pi^{\text{DEF}} \textit{ and an assignment}\\ &\qquad\qquad \theta : \text{VAR}(A) \longrightarrow \text{TERMS}\big(\text{ATOMS}(\Pi)\big)\ \big\} \cup\\ &\big\{\ \langle A\theta,+\rangle \ \big|\ \textit{there exist a constraint ``}\bot \leftarrow \mathsf{not}\, A\textit{'' } \in \Pi^{\text{DEF}},\ \big\}; \qquad (14)\end{aligned}$$

$$\begin{aligned}\mathcal{L}^{k+1}(\Pi) = &\mathcal{L}^k(\Pi) \cup\\ &\Big\{\ \langle A_1\theta_1,+\rangle \ \Big|\ \textit{there exist rules ``}A_1 \leftarrow B_1, \widehat{Bd_1}\textit{''},\ \textit{``}A_2 \leftarrow \mathsf{not}\, B_2, \widehat{Bd_2}\textit{'' } \in \Pi^{\text{DEF}}\\ &\qquad\qquad \textit{and assignments } \theta_i : \text{VAR}\big(\{A_i,B_i\}\big) \longrightarrow \text{TERMS}\big(\mathcal{L}^k(\Pi)\big)\ (i=1,2),\\ &\qquad\qquad \textit{such that:}\\ &\qquad\qquad \textit{(1) } A_1\theta_1 = A_2\theta_2 \textit{ and } (B_1\theta_1 = B_2\theta_2 \textit{ or } \{B_1,B_2\} = \emptyset);\\ &\qquad\qquad \textit{(2) } \widehat{Bd_1}\theta_1 \subseteq \mathcal{L}^k(\Pi) \textit{ and } \widehat{Bd_2}\theta_2 \subseteq \mathcal{L}^k(\Pi)\ \Big\} \cup \qquad (15)\end{aligned}$$

$$\begin{aligned}&\Big\{\ \langle A_1\theta_1,-\rangle \ \Big|\ \textit{there exist rules ``}A_1' \leftarrow A_1, B_1, \widehat{Bd_1}\textit{''},\ \textit{``}A_2' \leftarrow A_2, \mathsf{not}\, B_2, \widehat{Bd_2}\textit{''}\\ &\qquad\qquad \in \Pi^{\text{DEF}} \textit{ and assignments } \theta_i : \text{VAR}\big(\{A_i,B_i\}\big) \longrightarrow \text{TERMS}\big(\mathcal{L}^k(\Pi)\big)\\ &\qquad\qquad (i=1,2), \textit{such that:}\\ &\qquad\qquad \textit{(1) } A_1\theta_1 = A_2\theta_2 \textit{ and } (B_1\theta_1 = B_2\theta_2 \textit{ or } \{B_1,B_2\} = \emptyset);\\ &\qquad\qquad \textit{(2) } \widehat{Bd_1}\theta_1 \subseteq \mathcal{L}^k(\Pi) \textit{ and } \widehat{Bd_2}\theta_2 \subseteq \mathcal{L}^k(\Pi);\\ &\qquad\qquad \textit{(3) } A_1'\theta_1 \in \mathcal{L}^k(\Pi)^- \textit{ and } A_2'\theta_2 \in \mathcal{L}^k(\Pi)^-\ \Big\} \cup \qquad (16)\end{aligned}$$

4. For ease of presentation in the following, for some given rules "$A_1 \leftarrow B_1, \widehat{Bd_1}$","$A_2 \leftarrow \mathsf{not}\, B_2, \widehat{Bd_2}$" $\in \Pi^{\text{DEF}}$, we denote by $\{B_1,B_2\} = \emptyset$ the possibility that both B_1 and B_2 does not actually exists in the description "$A_1 \leftarrow B_1, \widehat{Bd_1}$" and "$A_2 \leftarrow \mathsf{not}\, B_2, \widehat{Bd_2}$" of the two rules. This saves having to write the other case when we actual have just "$A_1 \leftarrow \widehat{Bd_1}$" and "$A_2 \leftarrow \widehat{Bd_2}$", respectively.

$$\Big\{ \langle A_1\theta_1, + \rangle \;\Big|\; \textit{there exist rules } \text{“}A_1' \leftarrow \mathsf{not}\, A_1, B_1, \widehat{Bd_1}\text{”},\ \text{“}A_2' \leftarrow \mathsf{not}\, A_2, \mathsf{not}\, B_2, \widehat{Bd_2}\text{”}$$
$$\in \Pi^{\text{DEF}} \textit{ and assignments } \theta_i : \textsc{var}\big(\{A_i, B_i\}\big) \longrightarrow \textsc{terms}\big(\mathcal{L}^k(\Pi)\big)$$
$$(i = 1, 2),\ \textit{such that:}$$
$$\textit{(1) } A_1\theta_1 = A_2\theta_2 \textit{ and } (B_1\theta_1 = B_2\theta_2 \textit{ or } \{B_1, B_2\} = \emptyset);$$
$$\textit{(2) } \widehat{Bd_1}\theta_1 \subseteq \mathcal{L}^k(\Pi) \textit{ and } \widehat{Bd_2}\theta_2 \subseteq \mathcal{L}^k(\Pi);$$
$$\textit{(3) } A_1'\theta_1 \in \mathcal{L}^k(\Pi)^- \textit{ and } A_2'\theta_2 \in \mathcal{L}^k(\Pi)^- \Big\}; \tag{17}$$
$$\mathcal{L}^\infty(\Pi) = \bigcup_{i=0}^{\infty} \mathcal{L}^i(\Pi),$$

where for $\odot \in \{+, -\}$ *and* $S \subseteq \mathcal{L}^\infty(\Pi)$*, we denote by* $S^\odot$ *as the set* $\big\{p(\mathbf{t}) \mid \langle p(\mathbf{t}), \odot\rangle \in S\big\}$ *(as such, the sets* $\mathcal{L}^k(\Pi)^-$ *stands for the definitely false atoms:* $\big\{p(\mathbf{t}) \mid \langle p(\mathbf{t}), -\rangle \in \mathcal{L}^k(\Pi)^-\big\}$*)*[5].

In (15), (16) and (17), we further denote $\widehat{Bd_i}$ *(*$i = 1, 2$*) as the remaining body atoms of the rules* “$A_i' \leftarrow A_i, B_i, \widehat{Bd_i}$”*, and where by the expression* “$\widehat{Bd_i}\theta_i \subseteq \mathcal{L}^k(\Pi)$”*, we mean that for all positive and negative atoms* $A_i''\theta_i$ *and* $\mathsf{not}\, B_i''\theta_i$ *in* $\widehat{Bd_i}\theta_i$*,* $A_i''\theta_i \in \mathcal{L}^k(\Pi)^+$ *and* $B_i''\theta_i \in \mathcal{L}^k(\Pi)^-$*.*

Here we view the two symbols: “$\top$*” and “*$\bot$*”, as mentioned in (13), as the* interpreted *propositional atoms* truth *and* falsity*, respectively (and whose interpretation is self-explanatory). For convenience, we will omit explicitly writing that “*$\langle \top, + \rangle$*” and “*$\langle \bot, - \rangle$*” are in the set* $\mathcal{L}^0(\Pi) \subseteq \mathcal{L}^\infty(\Pi)$ *since we will assume it from the context.*

Let us take a closer look at Definition 2. Generally speaking, $\mathcal{L}^\infty(\Pi)$ induces two subsets of atoms of Π which, by instantiating with their ground instances from the input databases, are known to be *definitely true* and *definitely false*, as contained in $\mathcal{L}^\infty(\Pi)^+$ and $\mathcal{L}^\infty(\Pi)^-$, respectively. Indeed, the base case (14) of the forms $\langle A, + \rangle$ and $\langle A\theta, \odot\rangle$ $\big($where $\odot \in \{+, -\}\big)$ are derived from the facts, and positive and negative atomic constraints of the program Π, respectively. Then the sets (15), (16) and (17) as specified in the inductive step, represent the propagations of the positive and negative facts, respectively, using facts already derived from the previous stages. For instance, let us consider (15), the tuple $\langle A_1\theta_1, + \rangle$ is derived from the rules $A_1 \leftarrow B_1, \widehat{Bd_1}$ and $A_2 \leftarrow \mathsf{not}\, B_2, \widehat{Bd_2} \in \Pi^{\text{DEF}}$ and the assignments θ_1 and θ_2, under the two conditions (1) and (2) as described in the definition, i.e., (15). These conditions allow the resolution rule to be used to derive the fact $A_1\theta_1$ from Π. A similar concept applies to the derivation of the negative fact $\langle A_1\theta_1, - \rangle$ as described in (16), and the other positive facts from (17).

Finally, we further note that it can be the case that $\mathcal{L}^k(\Pi)^+ \cap \mathcal{L}^k(\Pi)^- \neq \emptyset$, which implies that the program Π is inconsistent.

5. In this paper, we use $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, etc., (and possibly with subscripts) to denote tuples of variables only, while we use $\mathbf{t}$, $\mathbf{u}$, $\mathbf{v}$, etc., to denote tuples of arbitrary terms.

Example 3. Let Π be a program consisting of the following rules:

$$
\begin{aligned}
\mathsf{r}_1 :&\ \ \bot \leftarrow \mathsf{q}(X),\\
\mathsf{r}_2 :&\ \ \mathsf{p}\big(f(X)\big) \leftarrow \mathsf{p}(X),\, \mathsf{r}(X),\, \mathsf{not}\, \mathsf{q}(X),\\
\mathsf{r}_3 :&\ \ \mathsf{r}(X) \leftarrow \mathsf{p}(X),\, \mathsf{r}\big(f(X)\big),\\
\mathsf{r}_4 :&\ \ \bot \leftarrow \mathsf{r}(X),\, \mathsf{p}\big(g(X)\big),\, \mathsf{not}\ \mathsf{q}(X),\\
\mathsf{r}_5 :&\ \ \bot \leftarrow \mathsf{r}(Y),\, \mathsf{not}\ \mathsf{p}\big(g(Y)\big),\, \mathsf{not}\, \mathsf{q}(Y).
\end{aligned}
$$

Then we have that $\textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}})\big) = \big\{X,\ Y,\ f(X),\ g(X),\ g(Y)\big\}$ (please see section 2.1 for the notion of $\textsc{terms}(S)$, for some set of atoms S), i.e., all the possible terms and sub-terms mentioned in any atom of $\Pi^{\textsc{def}}$. Then according to (14) of Definition 2, we get for the first step, $\mathcal{L}^0(\Pi)$:

$$
\mathcal{L}^0(\Pi) = \Big\{\langle \mathsf{q}(X), -\rangle,\, \langle \mathsf{q}(Y), -\rangle,\, \big\langle \mathsf{q}\big(f(X)\big), -\big\rangle,\, \big\langle \mathsf{q}\big(g(X)\big), -\big\rangle,\, \big\langle \mathsf{q}\big(g(Y)\big), -\big\rangle\Big\},
$$

because we have from (14) that

$$
\mathcal{L}^0(\Pi) = \Big\{\langle \mathsf{q}(X)\theta, -\rangle \ \Big|\ \mathsf{r}_1 = \text{“}\bot \leftarrow \mathsf{q}(X)\text{”} \in \Pi^{\textsc{def}} \text{ and } \theta : \{X\} \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}})\big)\Big\}.
$$

Then for the next step $\mathcal{L}^1(\Pi)$, we get that

$$
\begin{aligned}
&\mathcal{L}^1(\Pi) = \mathcal{L}^0(\Pi) \cup \\
&\Big\{\big\langle \mathsf{r}(X), - \big\rangle \;\big|\; \mathsf{r}_4 = \text{“}\bot \leftarrow \mathsf{r}(X),\ \mathsf{p}\big(g(X)\big),\ \mathsf{not}\ \mathsf{q}(X)\text{”} \in \Pi^{\text{DEF}},\ \theta_1 = \{X \mapsto X\}, \qquad (18)\\
&\qquad\qquad \mathsf{r}_5 = \text{“}\bot \leftarrow \mathsf{r}(Y),\ \mathsf{not}\ \mathsf{p}\big(g(Y)\big),\ \mathsf{not}\ \mathsf{q}(Y)\text{”} \in \Pi^{\text{DEF}},\ \theta_2 = \{Y \mapsto X\}, \qquad (19)\\
&\qquad\qquad \mathsf{q}(X)\theta_1 = \mathsf{q}(Y)\theta_2 \in \mathcal{L}^0(\Pi)^- \text{ and } \mathsf{p}\big(g(X)\big)\theta_1 = \mathsf{p}\big(g(Y)\big)\theta_2\Big\} \cup \\
&\Big\{\big\langle \mathsf{r}(Y), - \big\rangle \;\big|\; \mathsf{r}_4 = \text{“}\bot \leftarrow \mathsf{r}(X),\ \mathsf{p}\big(g(X)\big),\ \mathsf{not}\ \mathsf{q}(X)\text{”} \in \Pi^{\text{DEF}},\ \theta_1 = \{X \mapsto Y\},\\
&\qquad\qquad \mathsf{r}_5 = \text{“}\bot \leftarrow \mathsf{r}(Y),\ \mathsf{not}\ \mathsf{p}\big(g(Y)\big),\ \mathsf{not}\ \mathsf{q}(Y)\text{”} \in \Pi^{\text{DEF}},\ \theta_2 = \{Y \mapsto Y\},\\
&\qquad\qquad \mathsf{q}(X)\theta_1 = \mathsf{q}(Y)\theta_2 \in \mathcal{L}^0(\Pi)^- \text{ and } \mathsf{p}\big(g(X)\big)\theta_1 = \mathsf{p}\big(g(Y)\big)\theta_2\Big\} \cup \\
&\Big\{\big\langle \mathsf{r}\big(f(X)\big), - \big\rangle \;\big|\; \mathsf{r}_4 = \text{“}\bot \leftarrow \mathsf{r}(X),\ \mathsf{p}\big(g(X)\big),\ \mathsf{not}\ \mathsf{q}(X)\text{”} \in \Pi^{\text{DEF}},\ \theta_1 = \big\{X \mapsto f(X)\big\},\\
&\qquad\qquad \mathsf{r}_5 = \text{“}\bot \leftarrow \mathsf{r}(Y),\ \mathsf{not}\ \mathsf{p}\big(g(Y)\big),\ \mathsf{not}\ \mathsf{q}(Y)\text{”} \in \Pi^{\text{DEF}},\ \theta_2 = \big\{Y \mapsto f(X)\big\},\\
&\qquad\qquad \mathsf{q}(X)\theta_1 = \mathsf{q}(Y)\theta_2 \in \mathcal{L}^0(\Pi)^- \text{ and } \mathsf{p}\big(g(X)\big)\theta_1 = \mathsf{p}\big(g(Y)\big)\theta_2\Big\} \cup \\
&\Big\{\big\langle \mathsf{r}\big(g(X)\big), - \big\rangle \;\big|\; \mathsf{r}_4 = \text{“}\bot \leftarrow \mathsf{r}(X),\ \mathsf{p}\big(g(X)\big),\ \mathsf{not}\ \mathsf{q}(X)\text{”} \in \Pi^{\text{DEF}},\ \theta_1 = \big\{X \mapsto g(X)\big\},\\
&\qquad\qquad \mathsf{r}_5 = \text{“}\bot \leftarrow \mathsf{r}(Y),\ \mathsf{not}\ \mathsf{p}\big(g(Y)\big),\ \mathsf{not}\ \mathsf{q}(Y)\text{”} \in \Pi^{\text{DEF}},\ \theta_2 = \big\{Y \mapsto g(X)\big\},\\
&\qquad\qquad \mathsf{q}(X)\theta_1 = \mathsf{q}(Y)\theta_2 \in \mathcal{L}^0(\Pi)^- \text{ and } \mathsf{p}\big(g(X)\big)\theta_1 = \mathsf{p}\big(g(Y)\big)\theta_2\Big\} \cup \\
&\Big\{\big\langle \mathsf{r}\big(g(Y)\big), - \big\rangle \;\big|\; \mathsf{r}_4 = \text{“}\bot \leftarrow \mathsf{r}(X),\ \mathsf{p}\big(g(X)\big),\ \mathsf{not}\ \mathsf{q}(X)\text{”} \in \Pi^{\text{DEF}},\ \theta_1 = \big\{X \mapsto g(Y)\big\},\\
&\qquad\qquad \mathsf{r}_5 = \text{“}\bot \leftarrow \mathsf{r}(Y),\ \mathsf{not}\ \mathsf{p}\big(g(Y)\big),\ \mathsf{not}\ \mathsf{q}(Y)\text{”} \in \Pi^{\text{DEF}},\ \theta_2 = \big\{Y \mapsto g(Y)\big\},\\
&\qquad\qquad \mathsf{q}(X)\theta_1 = \mathsf{q}(Y)\theta_2 \in \mathcal{L}^0(\Pi)^- \text{ and } \mathsf{p}\big(g(X)\big)\theta_1 = \mathsf{p}\big(g(Y)\big)\theta_2\Big\},
\end{aligned}
$$

for which, we finally get that $\mathcal{L}^2(\Pi) = \mathcal{L}^1(\Pi) = \mathcal{L}^\infty(\Pi)$, i.e., we cannot infer any more certainly true or false atoms.

In particular, it is observed how we derive the (definitely false) atoms of the form “$\mathsf{r}(X)$”, in the step $\mathcal{L}^1(\Pi)$, through the use of the *resolution rule*. For instance, if we consider the constraints r_4 and r_5 under the respective assignments $\theta_1 = \{X \mapsto X\}$ and $\theta_2 = \{Y \mapsto X\}$ in the lines marked by (18) and (19) above, then we get that $\mathsf{r}_4\theta_1 =$ “$\bot \leftarrow \mathsf{r}(X),\ \mathsf{p}\big(g(X)\big),\ \mathsf{not}\ \mathsf{q}(X)$” and $\mathsf{r}_5\theta_2 =$ “$\bot \leftarrow \mathsf{r}(X),\ \mathsf{not}\ \mathsf{p}\big(g(X)\big),\ \mathsf{not}\ \mathsf{q}(X)$”, respectively.

Since we already know that the negative body atom $\mathsf{q}(X)$ of $\mathsf{r}_4\theta_1$ and $\mathsf{r}_5\theta_2$ is certainly false due to that we have obtained $\langle \mathsf{q}(X), - \rangle$ in the previous step $\mathcal{L}^0(\Pi)$, it follows that we can reduce the two constraints $\mathsf{r}_4\theta_1$ and $\mathsf{r}_5\theta_2$ into $\mathsf{r}_4^* =$ “$\bot \leftarrow \mathsf{r}(X),\ \mathsf{p}\big(g(X)\big)$” and $\mathsf{r}_5^* =$ “$\bot \leftarrow \mathsf{r}(X),\ \mathsf{not}\ \mathsf{p}\big(g(X)\big)$”, respectively.

Then loosely speaking, because r_4^* and r_5^* are now in the form “$\bot \leftarrow A,\ B$” and “$\bot \leftarrow A,\ \mathsf{not}\ B$”, respectively, we can finally apply the resolution rule to infer that the atom $\mathsf{r}(X)$ must be definitely false as well. Therefore, the pair $\langle \mathsf{r}(X), - \rangle$ is included in $\mathcal{L}^1(\Pi)$. □

Proposition 1. ***[Monotonicity]*** *For a given program* Π*, we have that for all* $0 \leq i \leq j$*,* $\mathcal{L}^i(\Pi) \subseteq \mathcal{L}^j(\Pi)$*.*

Proposition 2. ***[Lower bound for stable models]*** *Let* Π *be a program. Then for every input database* D *and stable model* M *of* $\Pi \cup D$*, we have that* $\mathcal{L}^\infty(\Pi)^+\restriction_{\text{CONST}(\Pi \cup D)} \subseteq M$ *and for each* $\alpha \in \mathcal{L}^\infty(\Pi)^-\restriction_{\text{CONST}(\Pi \cup D)}$*,* $\alpha \notin M$[6]*.*

Proof. For a given input database D, if the program $\Pi \cup D$ is inconsistent, then the statement is clearly true. Now we consider the case that the program $\Pi \cup D$ is consistent, and hence it has a stable model M. We prove the result by induction on $\mathcal{L}^k(\Pi)$ for $k \geq 0$. Indeed, the base case clearly holds when $k = 0$ because $\mathcal{L}^0(\Pi)\restriction_{\text{CONST}(\Pi \cup D)}$ corresponds to the grounding of both the positive (i.e., rules "$A \leftarrow \top$" or constraints "$\bot \leftarrow \mathsf{not}\, A$") and negative facts (i.e., constraints "$\bot \leftarrow A$") from Π^{DEF}. Suppose that the result also holds for $k > 0$. Now we consider the case $k + 1$. From Definition 2, we know that a new atom $A_1\theta_1$ is derived in $\mathcal{L}^{k+1}(\Pi)^+$ based on two cases of (15) and (17) in the definition, respectively. We first consider case (15). Obviously, the new atom $A_1\theta_1$ is derived based on two rules from Π^{DEF}: $A_1 \leftarrow B_1, \widehat{Bd_1}$ and $A_2 \leftarrow \mathsf{not}\, B_2, \widehat{Bd_2}$. From the conditions illustrated in (15) and the fact that $\mathcal{L}^k(\Pi)^+\restriction_{\text{CONST}(\Pi \cup D)} \subseteq M$, and $\forall A \in \mathcal{L}^k(\Pi)^-\restriction_{\text{CONST}(\Pi \cup D)}$, $A \notin M$, we can easily obtain that $A_1\theta_1 \notin M$. A similar argument is also for the case of (17).

On the other hand, for each new atom $A_1\theta_1$ in $\mathcal{L}^{k+1}(\Pi)^-\restriction_{\text{CONST}(\Pi \cup D)}$ derived from (16) in Definition 2, in a similar way, we can also show that $A_1\theta_1 \notin M$. □

3.2 Deriving Upper Bound

Based on Definition 2, now we define an upper bound for a given program Π as follows.

Definition 3. ***[Deriving upper bound]*** *Let* Π *be a program,* Π^{DEF} *be its definite normal form,* $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$*, ...,* $\text{SCC}(\Pi^{\text{DEF}})^{[K]}$ *be the SCC stratification of* $\text{SCC}(\Pi^{\text{DEF}})$ *(see Section 2.7) and* $S \subseteq \mathcal{L}^\infty(\Pi)$*. Then we define* $\mathcal{U}^i(\Pi, S)$ *(*$i \geq 0$*) inductively based on the three subcases: (1)* $i = 0$*; (2)* $1 \leq i \leq K$*; and (3)* $i > K$*, as follows*[7]*:*

$i = 0$:

$$\begin{aligned}\mathcal{U}^0(\Pi, S) = \big\{\, Hd(r)\theta \mid\ & \textit{there exist a rule } r \in \text{SCC}(\Pi^{\text{DEF}})^{[0]} \textit{ and an assignment} \\ & \theta : \text{VAR}(r) \longrightarrow \text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}})\big) \textit{ such that:} \\ & \textit{(1) } \text{PRED}\big(Pos(r)\big) \cap \text{INT}(\Pi) = \emptyset; \\ & \textit{(2) } Pos(r)\theta \cap S^- = \emptyset \textit{ and } Neg(r)\theta \cap S^+ = \emptyset \,\big\}; \end{aligned} \tag{20}$$

6. Recall from Section 2.1 that for a set of atoms S and set of constants C, $S\restriction_C$ denotes the grounding of S under the constants mentioned in C.
7. See Section 2.7 for the definitions of $domPos(r)$ and $recrPos(r)$ positive body atoms of a rule $r \in \Pi^{\text{DEF}}$.

$1 \leq i \leq K$:

$$\begin{aligned}\mathcal{U}^i(\Pi, S) =& \mathcal{U}^{i-1}(\Pi, S) \cup \\ & \big\{ Hd(r)\theta \mid \textit{there exist a rule } r \in \textsc{scc}(\Pi^{\textsc{def}})^{[i]} \textit{ and an assignment} \\ & \qquad \theta : \textsc{var}(r) \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}}) \cup \mathcal{U}^{i-1}(\Pi, S)\big) \textit{ such that:} \\ & \qquad \textit{(1) } \textsc{terms}\big(extPos(r)\theta\big) \subseteq \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}})\big); \\ & \qquad \textit{(2) } domPos(r)\theta \subseteq \mathcal{U}^{i-1}(\Pi, S); \\ & \qquad \textit{(3) } Pos(r)\theta \cap S^- = \emptyset \textit{ and } Neg(r)\theta \cap S^+ = \emptyset \,\big\}; \end{aligned} \tag{21}$$

$i > K$:

$$\begin{aligned}\mathcal{U}^i(\Pi, S) =& \mathcal{U}^{i-1}(\Pi, S) \cup \\ & \big\{ Hd(r)\theta \mid \textit{there exist a rule } r \in \bigcup_{j=0}^{j=K} \textsc{scc}(\Pi^{\textsc{def}})^{[j]} \textit{ and an assignment} \\ & \qquad \theta : \textsc{var}(r) \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}}) \cup \mathcal{U}^{i-1}(\Pi, S)\big) \textit{ such that:} \\ & \qquad \textit{(1) } \textsc{terms}\big(extPos(r)\theta\big) \subseteq \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}})\big); \\ & \qquad \textit{(2) } domPos(r)\theta \subseteq \mathcal{U}^{K}(\Pi, S); \\ & \qquad \textit{(3) } recrPos(r)\theta \subseteq \mathcal{U}^{i-1}(\Pi, S); \\ & \qquad \textit{(4) } Pos(r)\theta \cap S^- = \emptyset \textit{ and } Neg(r)\theta \cap S^+ = \emptyset \,\big\}. \end{aligned} \tag{22}$$

Finally, we define $\mathcal{U}^\infty(\Pi, S) = \bigcup_{i=0}^{\infty} \mathcal{U}^i(\Pi, S)$ *to be its fixpoint.*

Now we take a closer look at Definition 3. In a nutshell, we define the upper bound $\mathcal{U}^i(\Pi, S)$ $(i \geq 0)$ inductively through three subcases: (1) $i = 0$; (2) $1 \leq i \leq K$; and (3) $i > K$, which we explain as follows:

$i = 0$: This initial step first considers rules $r \in \textsc{scc}(\Pi^{\textsc{def}})^{[0]}$, which are actually those rules $r \in \Pi^{\textsc{def}}$, where both the classes of positive body atoms $domPos(r)$ and $recrPos(r)$ are empty, i.e., $\textsc{pred}\big(Pos(r)\big) \cap \textsc{int}(\Pi^{\textsc{def}}) = \emptyset$. Intuitively, this first step corresponds to deriving the atoms of rules that do not depend on the other rules of $\Pi^{\textsc{def}}$, apart from database facts. Also, we note that because we only consider rules from the lowest stratum $\textsc{scc}(\Pi^{\textsc{def}})^{[0]}$, then rules $r' \in \textsc{scc}(\Pi^{\textsc{def}})^{[0]}$ for which $\textsc{pred}\big(Pos(r)\big) \cap \textsc{int}(\Pi^{\textsc{def}}) \neq \emptyset$ will not actual "fire" under any assignment θ because $\big($based on our definition of the SCC stratification of $\textsc{scc}(\Pi^{\textsc{def}})\big)$, the fact that r' is in the lowest stratum $\big($i.e., $\textsc{scc}(\Pi^{\textsc{def}})^{[0]}\big)$ implies that there are no rules in $\Pi^{\textsc{def}}$ whose head unifies with any of the positive body atoms $domPos(r') \cup recrPos(r')$ of r';

$1 \leq i \leq K$: Then based on the atoms derived from the initial step $\mathcal{U}^0(\Pi, S)$, we inductively define the steps $\mathcal{U}^i(\Pi, S)$ (for $i \in \{1, \ldots, K\}$) based on the SCC stratification

of $\text{SCC}(\Pi^{\text{DEF}})$. Indeed, at each step $\mathcal{U}^i(\Pi, S)$, we consider the heads of those rules r in the stratum $\text{SCC}(\Pi^{\text{DEF}})^{[i]}$ of the SCC stratification $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$, ..., $\text{SCC}(\Pi^{\text{DEF}})^{[K]}$ of Π^{DEF}, for which, their domain (positive) body atoms $domPos(r)$ have already been established from the previous steps, i.e., $domPos(r)\theta \subseteq \mathcal{U}^{i-1}(\Pi, S)$ for some assignment θ: $\text{VAR}(r) \longrightarrow \text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}}) \cup \mathcal{U}^{i-1}(\Pi, S)\big)$. Here, we note that the assignment θ considers all the possible terms as well as subterms (see Section 2.1 for the definition of $\text{TERM}(t)$ which considers all the subterms of a term t). In addition, we also note that the extension body atoms in $extPos(r)$ are only restricted to mention terms from $\text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}})\big)$. In fact, even throughout the remaining steps, we will fix this restriction on the extensional atoms $extPos(r)$, for any rule r because these extensional atoms will not play in the further growth of complex terms.

$i > K$: Finally, having already established the derivation of possible head relation types of the rules in $\text{SCC}(\Pi^{\text{DEF}})^{[j]}$ (for $j \in \{1, \ldots, K\}$), the remaining steps where $i > K$ (which can be infinite) further consider the applications of the rules based on recursion.

Assuming that we have already derived $\mathcal{U}^{i-1}(\Pi, S)$, for $i > K$, then we further derive $\mathcal{U}^i(\Pi, S)$ based on $\mathcal{U}^{i-1}(\Pi, S)$ as follows: for each rule $r \in \bigcup_{j=0}^{j=K} \text{SCC}(\Pi^{\text{DEF}})^{[j]}$ (i.e., the union of all SCC stratum) and each assignment θ: $\text{VAR}(r) \longrightarrow \text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}}) \cup \mathcal{U}^{i-1}(\Pi, S)\big)$ where the following four conditions hold:

$$(1)\ \text{TERMS}\big(extPos(r)\theta\big) \subseteq \text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}})\big); \tag{23}$$

$$(2)\ domPos(r)\theta \subseteq \mathcal{U}^K(\Pi, S); \tag{24}$$

$$(3)\ recrPos(r)\theta \subseteq \mathcal{U}^{i-1}(\Pi, S); \tag{25}$$

$$(4)\ Pos(r)\theta \cap S^- = \emptyset \text{ and } Neg(r)\theta \cap S^+ = \emptyset, \tag{26}$$

we add $Hd(r)\theta$ into the next stage $\mathcal{U}^i(\Pi, S)$. In particular, note that we only limit θ to map the variables of r to the terms of Π^{DEF} and that of the earlier steps $\mathcal{U}^j(\Pi, S)\big)$ $(0 \leq j \leq i-1)$. As already mentioned and explained above, the first of these conditions, (23) enforces that the atoms of extensional relations in $extPos(r)$ can only be mapped to terms from $\text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}})\big)$. The next condition (24) enforces that the domain atoms in $domPos(r)$ are confined only to the K^{th}-step $\mathcal{U}^K(\Pi, S)$. Similarly to the case for the atoms in $extPos(r)$, we restrict the atoms in $domPos(r)$ because they will not really play a part in the further propagation of complex terms through recursion. In contrast, the third condition (25) now allows those (mutually) recursive atoms in $recrPos(r)$ to finally consider any complex term that can grow within the steps of $\mathcal{U}^{i-1}(\Pi, S)$. Finally, the last condition (26) enforces us to only consider r if it is also consistent with the subset $S \subseteq \mathcal{L}^\infty(\Pi)$ of the lower bound.

We further note that for any of the three cases: (1) $i = 0$; (2) $1 \leq i \leq K$; and (3) $i > K$ above, we only derive the head $Hd(r)\theta$ if it is also the case that (under the assignment θ) both the conditions $Pos(r)\theta \cap S^- = \emptyset$ and $Neg(r)\theta \cap S^+ = \emptyset$. These aforementioned conditions enforce us to only consider rules that will be consistent with the subset $S \subseteq \mathcal{L}^\infty(\Pi)$ of the lower bound. Figure 4 provides a conceptual diagram of the ideas behind the upper bound as specified through Definition 3.

Proposition 3. ***[Monotonicity I]*** *For all* $0 \leq i \leq j$ *and* $S \subseteq \mathcal{L}^\infty(\Pi)$, $\mathcal{U}^i(\Pi, S) \subseteq \mathcal{U}^j(\Pi, S)$.

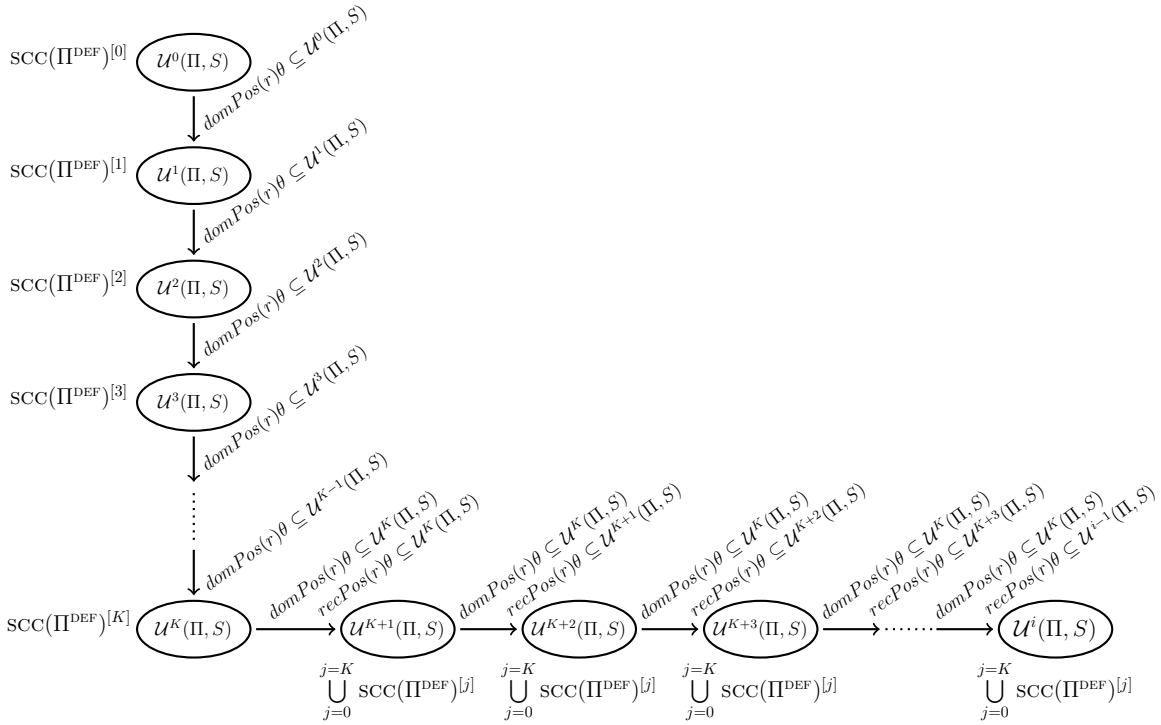

Figure 4: Conceptualization of the upper bound $\mathcal{U}(\Pi, S)^i$ as specified in Definition 3.

Proposition 4. ***[Monotonicity II]*** *For all $i \geq 0$ and S_1, $S_2 \subseteq \mathcal{L}^\infty(\Pi)$, $S_1 \subseteq S_2$ implies $\mathcal{U}^i(\Pi, S_2) \subseteq \mathcal{U}^i(\Pi, S_1)$.*

Proof. With respect to (20), (21) and (22) of Definition 3 above, if $S_1 \subseteq S_2$, then $Pos(r)\theta \cap S_2^- = \emptyset$ $\big(Neg(r)\theta \cap S_2^+ = \emptyset$, resp.$\big)$ implies $Pos(r)\theta \cap S_1^- = \emptyset$ $\big(Neg(r)\theta \cap S_1^+ = \emptyset$, resp.$\big)$. □

Proposition 5. ***[Finiteness]*** *For a program Π and any $S \subseteq \mathcal{L}^\infty(\Pi)$, we have that $\mathcal{U}^i(\Pi, S)$ is a finite set for all $i \geq 0$.*

Proof. By induction. Clearly, based on (20) of Definition 3, we have that $\mathcal{U}^0(\Pi, S)$ is finite because there are only a finite number of rules in Π^{DEF} and only a finite number of terms (and subterms) in $\text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}})\big)$. Then assuming that $\mathcal{U}^j(\Pi, S)$ is finite for $1 \leq j \leq i$, it follows from the descriptions in (21)-(22) of Definition 3 that $\mathcal{U}^{i+1}(\Pi, S)$ must be a finite set as well because all the new atoms added to $\mathcal{U}^{j+1}(\Pi, S)$ are obtained only through assignments from terms in $\text{TERMS}\big(\mathcal{U}^j(\Pi, S)\big)$, which we already know to be finite. □

Theorem 1. ***[Upper bound for stable models]*** *Let Π be a program and $S \subseteq \mathcal{L}^\infty(\Pi)$. Then for every input database D and stable model M of $\Pi \cup D$, $M \ll \mathcal{U}^\infty(\Pi, S)$.*

Example 4. Let Π_i $(i \geq 0)$ be the programs defined as follows: for $i = 0$, we set:

$$\Pi_0 = \big\{ p_1\big(X, f(Y), f(Z)\big) \leftarrow p_0(X, Y, Z) \big\}.$$

While for $i > 0$, we set:

$$\Pi_i = \Big\{ p_i\big(f(X), Y, Z\big) \leftarrow p_i\big(X, f(Y), Z\big), \tag{27}$$

$$p_i\big(f(X), Y, Z\big) \leftarrow p_i\big(X, Y, f(Z)\big) \Big\}; \tag{28}$$

$$\Pi_{i+1} = \Big\{ p_{i+1}(Y, X, X) \leftarrow p_i(X, Y, Z) \Big\}, \tag{29}$$

where each p_i is a different predicate. Finally, for some given $k > 1$, we specify the program $\Pi[k] = \Pi_0 \cup \Pi_1 \cup \Pi_2 \cup \ldots \cup \Pi_{k-1} \cup \Pi_k$.

Now we show how we compute $\mathcal{U}^\infty\big(\Pi[k], \emptyset\big)$. Firstly, based on the notion of the SCCs of the firing graph $\Omega\big(\Pi[k]\big)$ of $\Pi[k]$ as we defined in Sections 2.6 and 2.7, we get that $\textsc{scc}\big(\Pi[k]\big) = \{\, \Pi_0,\ \Pi_1,\ \Pi_2,\ \ldots,\ \Pi_{k-1},\ \Pi_k \,\}$, and where: $\textsc{scc}\big(\Pi[k]\big)^{[0]} = \{\, \Pi_0 \}$, $\textsc{scc}\big(\Pi[k]\big)^{[1]} = \{\, \Pi_1 \}$, $\textsc{scc}\big(\Pi[k]\big)^{[2]} = \{\, \Pi_2 \}$, $\ldots$, $\textsc{scc}\big(\Pi[k]\big)^{[k-1]} = \{\, \Pi_{k-1} \}$, $\textsc{scc}\big(\Pi[k]\big)^{[k]} = \{\, \Pi_k \}$, forms the SCC stratification of $\textsc{scc}\big(\Pi[k]\big)$. Then based on (20) and (21) of Definition 3, we compute the first steps, $i = 0$ to $i = k$, of $\mathcal{U}^i\big(\Pi[k], \emptyset\big)$ inductively as follows:

$$\begin{aligned}
\mathcal{U}^0\big(\Pi[k], \emptyset\big) &= \Big\{ p_1\big(X, f(Y), f(Z)\big)\theta \;\Big|\; \text{“}p_1(X, Y, Z) \leftarrow p_0(X, Y, Z)\text{”} \in \Pi_0 \text{ and} \\
&\qquad\qquad \theta : \{X, Y, Z\} \longrightarrow \textsc{terms}\big(\textsc{atoms}\big(\Pi[k]\big)\big) \Big\}, \\
&= \Big\{ p_1\big(X, f(Y), f(Z)\big) \;\Big|\; X, Y, Z \in \big\{X, Y, Z, f(X), f(Y), f(Z)\big\} \Big\},
\end{aligned} \tag{30}$$

and for $i \in \{1, 2, \ldots, k-2, k-1\}$:

$$\begin{aligned}
\mathcal{U}^i\big(\Pi[k], \emptyset\big) &= \mathcal{U}^{i-1}\big(\Pi[k], \emptyset\big) \cup \\
&\quad \Big\{ p_i\big(f(X), Y, Z\big)\theta \;\Big|\; \text{“}p_i(X, Y, Z) \leftarrow p_i\big(X, f(Y), Z\big)\text{”} \in \Pi_i \text{ or} \\
&\qquad\qquad \text{“}p_i(X, Y, Z) \leftarrow p_i\big(X, Y, f(Z)\big)\text{”} \in \Pi_i \text{ and} \\
&\qquad\qquad \theta : \{X, Y, Z\} \longrightarrow \\
&\qquad\qquad\quad \textsc{terms}\big(\textsc{atoms}\big(\Pi[k]\big) \cup \mathcal{U}^{i-1}\big(\Pi[k], \emptyset\big)\big) \Big\} \\
&= \mathcal{U}^{i-1}\big(\Pi[k], \emptyset\big) \cup \\
&\quad \Big\{ p_i\big(f(X), Y, Z\big) \;\Big|\; \text{“}p_i(X, Y, Z) \leftarrow p_i\big(X, f(Y), Z\big)\text{”} \in \Pi_i \text{ or} \\
&\qquad\qquad \text{“}p_i(X, Y, Z) \leftarrow p_i\big(X, Y, f(Z)\big)\text{”} \in \Pi_i \text{ and} \\
&\qquad\qquad X, Y, Z \in \textsc{terms}\big(\textsc{atoms}\big(\Pi[k]\big) \cup \mathcal{U}^{i-1}\big(\Pi[k], \emptyset\big)\big) \Big\}
\end{aligned} \tag{31}$$

$$\begin{aligned}
\mathcal{U}^{i+1}\big(\Pi[k],\emptyset\big) \;=\; & \mathcal{U}^{i}\big(\Pi[k],\emptyset\big) \cup \\
& \Big\{ p_{i+1}(Y,X,X)\theta \;\Big|\; r = \text{“}p_{i+1}(Y,X,X) \leftarrow p_i(X,Y,Z)\text{”} \in \Pi_{i+1}, \\
& \qquad \theta : \{X,Y,Z\} \longrightarrow \\
& \qquad \textsc{terms}\big(\textsc{atoms}\big(\Pi[k]\big) \cup \mathcal{U}^{i}\big(\Pi[k],\emptyset\big)\big) \text{ and} \\
& \qquad \mathit{domPos}(r) = \big\{p_i(X,Y,Z)\theta\big\} \subseteq \mathcal{U}^{i}\big(\Pi[k],\emptyset\big) \Big\} \\
= \; & \mathcal{U}^{i-1}\big(\Pi[k],\mathbb{P}(\emptyset)\big) \cup \\
& \Big\{ p_{i+1}(Y,X,X) \;\Big|\; r = \text{“}p_{i+1}(Y,X,X) \leftarrow p_i(X,Y,Z)\text{”} \in \Pi_{i+1}, \\
& \qquad X,Y,Z \in \textsc{terms}\big(\textsc{atoms}\big(\Pi[k]\big) \cup \mathcal{U}^{i}\big(\Pi[k],\emptyset\big)\big) \text{ and} \\
& \qquad p_i(X,Y,Z) \in \mathcal{U}^{i}\big(\Pi[k],\emptyset\big) \Big\}, \qquad (32)
\end{aligned}$$

Then finally, we get from (22) of Definition 3 that for $i > k$, we get that:

$$\begin{aligned}
\mathcal{U}^{i}\big(\Pi[k],\emptyset\big) \;=\; & \mathcal{U}^{i-1}\big(\Pi[k],\emptyset\big) \cup \\
& \Big\{ p_i\big(f(X),Y,Z\big) \;\Big|\; r' = \text{“}p_i(X,Y,Z) \leftarrow p_i\big(X,f(Y),Z\big)\text{”} \in \Pi_i \text{ or} \\
& \qquad r' = \text{“}p_i(X,Y,Z) \leftarrow p_i\big(X,Y,f(Z)\big)\text{”} \in \Pi_i, \\
& \qquad X,Y,Z \in \textsc{terms}\big(\textsc{atoms}\big(\Pi[k]\big) \cup \mathcal{U}^{i-1}\big(\Pi[k],\emptyset\big)\big) \text{ and} \\
& \qquad \mathit{recrPos}(r)\theta \subseteq \mathcal{U}^{i-1}\big(\Pi[k],\emptyset\big) \Big\} \qquad (33)
\end{aligned}$$

$$\begin{aligned}
\mathcal{U}^{i+1}\big(\Pi[k],\emptyset\big) \;=\; & \mathcal{U}^{i}\big(\Pi[k],\emptyset\big) \cup \\
& \Big\{ p_{i+1}(Y,X,X) \;\Big|\; r = \text{“}p_{i+1}(Y,X,X) \leftarrow p_i(X,Y,Z)\text{”} \in \Pi_{i+1}, \\
& \qquad X,Y,Z \in \textsc{terms}\big(\textsc{atoms}\big(\Pi[k]\big) \cup \mathcal{U}^{i}\big(\Pi[k],\emptyset\big)\big) \text{ and} \\
& \qquad \mathit{domPos}(r)\theta \subseteq \mathcal{U}^{k}\big(\Pi[k],\emptyset\big) \Big\}. \qquad (34)
\end{aligned}$$

In particular, we note in (34) that because we limit $\mathit{domPos}(r)\theta$ to be only contained within the set $\mathcal{U}^{k}\big(\Pi[k],\emptyset\big)$ of the k^{th}-step $\big($because $\mathit{recrPos}(r) = \emptyset$ for the rule in (34)$\big)$, then the growth of terms that we consider for the steps $i > k$ is only through the recursive rules r' in (33) because $\mathit{recrPos}(r') \neq \emptyset$. It is for the condition that we confine the atoms of $\mathit{domPos}(r)\theta$ to those derived from the k^{th}-step $\mathcal{U}^{k}\big(\Pi[k],\emptyset\big)$ that all arguments of predicate symbols in program $\Pi[k]$ are limited. □

4. Polynomially Bounded Programs

Proposition 6. ***[Finite upper bound implies finite stable models]*** *For any program* Π *and any set* $S \subseteq \mathcal{L}^{\infty}(\Pi)$*, if* $\mathcal{U}^{\infty}(\Pi, S)$ *is a finite set, then* Π *is also finitely ground.*

Proof. Assume that for some given program Π and some set $S \subseteq \mathcal{L}^{\infty}(\Pi)$, we have that $\mathcal{U}^{\infty}(\Pi, S)$ is a finite set. Then clearly, since by Proposition 5 we have that each $\mathcal{U}^{i}(\Pi, S)$ is a finite set for all $i \geq 0$, then there must exists some $k \in \mathbb{N}$ such that $\mathcal{U}^{k}(\Pi, S) = \mathcal{U}^{\infty}(\Pi, S)$. Now let us assume without loss of generality that $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$, ..., $\text{SCC}(\Pi^{\text{DEF}})^{[K]}$ is the SCC stratification of the definite normal form Π^{DEF} of $\text{SCC}(\Pi^{\text{DEF}})$ (see Section 2.7).

Then with the SCC stratification: $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$, ..., $\text{SCC}(\Pi^{\text{DEF}})^{[K]}$ of the definite normal form Π^{DEF} of Π above, and for each $i = 0$ to $i = K$, let us denote by Π_i^{DEF} as the program containing the set of rules such that:

$$\Pi_i^{\text{DEF}} = \Big\{ r \,\Big|\, r \in \mathcal{C} \text{ and } \mathcal{C} \in \bigcup_{j=0}^{j=i} \text{SCC}(\Pi^{\text{DEF}})^{[j]} \Big\},$$

i.e., Π_i^{DEF} contains exactly all the rules mentioned in the SCCs from $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$ to $\text{SCC}(\Pi^{\text{DEF}})^{[i]}$. Then with the $k \in \mathbb{N}$ such that $\mathcal{U}^{k}(\Pi, S) = \mathcal{U}^{\infty}(\Pi, S)$ (i.e., as described above since $\mathcal{U}^{\infty}(\Pi, S)$ is a finite set and by Proposition 5), we will now show by induction on i that the program Π_i^{DEF} is limited (please see again Section 2.4 for the notion of *limited*).

Basis: $i = 0$:

Then from Definition 3, we have that $\mathcal{U}^{0}(\Pi, S)$ will be given by the set (20). Therefore, since no further additional atoms are derived for all the other stages $k > 0$ of $\mathcal{U}^{k}(\Pi, S)$ (i.e., since $\mathcal{U}^{k}(\Pi, S) = \mathcal{U}^{\infty}(\Pi, S)$), then it follows that the depth of terms in all the possible atoms that can be derived by the rules of Π_0^{DEF} are bounded and thus, all arguments are limited. Indeed, even because all the production of the mutually recursive head atom $\text{HD}(r)\theta$ with the body atoms $recPos(r)\theta$ (see (20) of Definition 3), of each rule $r \in \Pi_0^{\text{DEF}}$, are bounded (i.e., because $\mathcal{U}^{k}(\Pi, S) = \mathcal{U}^{\infty}(\Pi, S)$), then all the recursive arguments in $\text{ARG}(\Pi_0^{\text{DEF}})$ must be limited.

Inductive step: Now assume that Π_i^{DEF} is limited.

Then now let us consider Π_{i+1}^{DEF}. Then similarly to the base case above, since no further additional atoms are derived for all the stages $k > 0$ of $\mathcal{U}^{k}(\Pi, S)$, then it follows that the depth of terms mentioned in the recursive arguments corresponding to the mutually recursive head atom $\text{HD}(r)\theta$ with the body atoms $recPos(r)\theta$, of each rule $r \in \Pi_{i+1}^{\text{DEF}}$ (please see again (21)-(22) of Definition 3), are bounded and thus, are limited. On the other hand, since all the arguments in $\text{ARG}(\Pi_i^{\text{DEF}})$ are limited (ind. hyp.), then it follows that atoms in $domPos(r)\theta$ have arguments that are limited in Π_i^{DEF} so thus, all the arguments in $\text{ARG}(\Pi_{i+1}^{\text{DEF}})$ must therefore be limited.

Finally, the finite stable models for Π follows from the fact that all the arguments in $\text{ARG}(\Pi^{\text{DEF}})$ are limited. □

According to Proposition 6, it is clear that if $\mathcal{U}^{\infty}(\Pi, S)$ (for some $S \subseteq \mathcal{L}^{\infty}(\Pi)$) is a finite set, then Π is finitely ground. Also, from Definitions 2 and 3, we can see that if for each atom in $\mathcal{U}^{\infty}(\Pi, S)$, its term depth is bounded by a fixed integer, then $\mathcal{U}^{\infty}(\Pi, S)$ must be a finite set. So our attempt is to impose a bound $\mathcal{B}$ on $\text{DEP}\big(\mathcal{U}^{\infty}(\Pi, S)\big)$ such that $\text{DEP}\big(\mathcal{U}^{\mathcal{B}}(\Pi, S)\big) = \text{DEP}\big(\mathcal{U}^{\infty}(\Pi, S)\big)$. Based on this idea, we eventually are able to define a new class of programs called polynomially bounded programs by defining a term depth bound for $\mathcal{U}^{\mathcal{B}}(\Pi, S)$ to be a polynomial in the size of program Π.

Given $\text{DEP}\big(\mathcal{U}^{\mathcal{B}}(\Pi, S)\big) = \text{DEP}\big(\mathcal{U}^{\infty}(\Pi, S)\big)$, on the other hand, from Theorem 1, we know that for any $S \subseteq \mathcal{L}^{\infty}(\Pi)$, $\mathcal{U}^{\mathcal{B}}(\Pi, S)$ forms a finite upper bound for all stable models of Π. People may think that an easy way of getting such upper bound is simply to set $S = \emptyset$. However, from a computational viewpoint, we would prefer to have a relatively tight upper bound, instead of having an arbitrary polynomial upper bound. According to Proposition 4, this means that we should have a big set $S \subseteq \mathcal{L}^{\infty}(\Pi)$.

Firstly, given a program Π, we define

$$\mathbb{P}(\Pi) \;=\; N + N^3, \tag{35}$$

where $N = |\Pi^{\text{DEF}}| \cdot \text{maxArt}(\Pi) \cdot \text{maxPos}(\Pi) \cdot \text{maxDep}(\Pi)$, and such that

1. $\text{maxArt}(\Pi)$ denotes the product of the maximum arities of predicate and function symbols occurring in Π, i.e., $\text{maxArt}(\Pi) = m \times n$, where m and n are the maximum arities of the predicates and function symbols occurring in Π, respectively;

2. $\text{maxPos}(\Pi)$ denotes the maximum number of atoms in the positive body of a rule in Π;

3. $\text{maxDep}(\Pi)$ denotes the maximum depth of a term mentioned in an atom of Π, i.e.,

$$\text{maxDep}(\Pi) = \text{max}\big(\{\text{dep}(t_i) \,|\, p(t_1, \ldots, t_i, \ldots, t_n) \in \text{atoms}(\Pi)\}\big).$$

Intuitively, $\mathbb{P}(\Pi)$ gives an approximation of the minimum bound on the number of iterations of $\mathcal{U}^k(\Pi, S)$ that has to be done in order to determine if an infinite propagation of terms may actually take place. In a nutshell, iterating through $\mathcal{U}^k(\Pi, S)$, for $1 \leq k \leq \mathbb{P}(\Pi)$, considers all possible transpositions and propagations of an argument, say $p[i]$, within the program Π as we iterate through each step.

At the same time, such iteration will compute the maximum possible depth of any restricted arguments, as well as the possible "undoing" of these complex terms because we also incorporated the factor $\text{maxDep}(\Pi)$ into the number $\mathbb{P}(\Pi)$, which considers the maximum depth of a complex term mentioned in all the atoms of Π. Indeed, from the definition of $\mathbb{P}(\Pi)$ in (35), the factor $\text{maxArt}(\Pi)$ considers all possible transpositions of $p[i]$ within the arities of predicates and functions. In addition, the number $|\Pi^{\text{DEF}}| \cdot \text{maxPos}(\Pi)$ also factors in the possible transposition that can be propagated through each positive atom in the program.

In summary, the number of iterative steps $\mathbb{P}(\Pi)$ does three things: (i) the number $N = |\Pi^{\text{DEF}}| \cdot \text{maxArt}(\Pi) \cdot \text{maxPos}(\Pi) \cdot \text{maxDep}(\Pi)$ considers the iterative steps required to generate the deepest term of a restricted argument because it bounds the length of the longest possible path that can derive a complex term of a restricted argument; (ii) the number N^3 further adds the additional steps that are required to "undo" the complex terms compounded in the restricted arguments from doing the aforementioned first N-steps because it is the product of the number of the deepest possible term (bounded by N) with that of the maximum cycle length (also bounded by N) that can "undo" the complexity of the term and where the one more factor of N (which makes the term "cubed" in (35)) considers the possibility that each can take N-steps to exhaust each of the possible positions of arguments in atoms; and finally (iii) iterating $\mathbb{P}(\Pi)$-steps considers all possible

transpositions and propagations of an argument, which allows us to detect any growing cycles corresponding to an unlimited growth of complex terms within the argument, as well as to detect recursive information about function applications.

Example 5. Let Π_2 be a program consisting of the following rules:

$$\begin{aligned}
&\mathsf{r}_1 :\ \mathsf{p}_1\big(X, g(Y), h(Z)\big) \leftarrow \mathsf{p}_0\big(X, Y, Z\big),\\
&\mathsf{r}_2 :\ \mathsf{p}_2\big(X, g(Y), h(Z)\big) \leftarrow \mathsf{p}_1\big(X, Y, Z\big),\\
&\mathsf{r}_3 :\ \mathsf{p}_3\big(X, g(Y), h(Z)\big) \leftarrow \mathsf{p}_2\big(X, Y, Z\big),\\
&\mathsf{r}_4 :\ \mathsf{p}\big(X, Y, Z\big) \leftarrow \mathsf{p}_3\big(X, Y, Z\big),\\
&\mathsf{r}_5 :\ \mathsf{p}\big(f(X), Y, Z\big) \leftarrow \mathsf{p}\big(X, g(Y), Z\big),\\
&\mathsf{r}_6 :\ \mathsf{p}\big(f(X), Y, Z\big) \leftarrow \mathsf{p}\big(X, Y, h(Z)\big),
\end{aligned}$$

where "p_0" is an extensional predicate. Then it follows that all arguments apart from $p[1]$ are argument restricted. This is because, although the argumentation graph of Π_2 has a cycle about the arguments $\mathsf{p}[2]$ and $\mathsf{p}[3]$ (with themselves), they do not propagate the growth of complex terms. On the other hand, the argument $\mathsf{p}[1]$ is on a cycle propagating complex terms due to both of the rules r_5 and r_6 above. Now let $S = \emptyset$.

Then from rules r_1 - r_4, we would have that $\mathsf{p}\big(X, g^3(Y), h^3(Z)\big) \in \mathcal{U}^3(\Pi_2, S)$. Through the iterative applications of rules r_5 and r_6, we have that argument $\mathsf{p}[1]$ grows in depth through the function f. On the other hand, the depth that $\mathsf{p}[1]$ can grow is bounded by $k + l$ where k and l are the maximum depths reached by the arguments $\mathsf{p}[2]$ and $\mathsf{p}[3]$, respectively. In particular, we note that: (1) it takes 4-steps (via rules r_1 - r_4) to reach the maximum depth of the (restricted) arguments $\mathsf{p}[2]$ and $\mathsf{p}[3]$; (2) it takes a further 6-steps (via rules r_5 - r_6) to "undo" the built-up of those complex terms, and hence by the 10^{th}-step, $\mathcal{U}^{10}(\Pi_2, S) = \mathcal{U}^{\infty}(\Pi_2, S)$. Incidentally, it is not difficult to check that $10 < \mathbb{P}(\Pi_2)$. □

Let S be a set of atoms and Π a program. Then given an argument $p[i] \in \textsc{arg}(\Pi)$, we denote by $\textsc{dep}_{p[i]}\big(S\big)$ as the maximum term depth of the argument $p[i]$ as mentioned in some atom in S:

$$\textsc{dep}_{p[i]}\big(S\big) \ =\ \textsc{max}\big(\big\{\ \textsc{dep}(t_i)\ \mid\ p(t_1, \ldots, t_i, \ldots, t_n)\ \in\ S\big\}\big),$$

where we define $\textsc{dep}_{p[i]}\big(\emptyset\big) = 0$ when $S = \emptyset$. We are now ready to define the notion of polynomially limited arguments.

Definition 4. ***[Poly-limited arguments]*** *Let Π be a program and p a predicate in $\textsc{pred}(\Pi)$. We define $\textsc{poly}\mathrm{LA}_p(\Pi)$ to be the set of* poly-limited arguments *of p as follows:*

$$\Big\{\ p[i]\,\mid i \in \{1, \ldots, \textsc{arity}(p)\}\ \textit{and}\ \textsc{dep}_{p[i]}\big(\mathcal{U}^{\,2\,\cdot\,\mathbb{P}(\Pi)}(\Pi, S)\big)\ =\ \textsc{dep}_{p[i]}\big(\mathcal{U}^{\,3\,\cdot\,\mathbb{P}(\Pi)}(\Pi, S)\big)\ \Big\}, \quad (36)$$

where $S = \mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$.

Intuitively, $\textsc{poly}\mathrm{LA}_p(\Pi)$ denotes the set of arguments $p[i]$ $\big(i \in \big\{1, \ldots, \textsc{arity}(p)\big\}\big)$ that do not grow in depth beyond the stage of iterations $2 \cdot \mathbb{P}(\Pi)$ of $\mathcal{U}^k(\Pi, S)$, where S is the

set of atoms obtained from the lower bound $\mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$. Specifically, for an argument $p[i]$, we have that condition (36) holds for each SCC $\mathcal{C}$ of the activation graph $\Omega(\Pi)$, where $p \in \text{PRED}(\mathcal{C})$. Conceptually, we can think of the number of iterations $3 \cdot \mathbb{P}(\Pi)$ as three sections:

$\mathbb{P}(\Pi)$	$2 \cdot \mathbb{P}(\Pi)$	$3 \cdot \mathbb{P}(\Pi)$
generate atoms	initial depth	recursive depth

where the first section "$\mathbb{P}(\Pi)$" generates all the atoms (as will be revealed in Lemma 2 in Appendix B.3); the next section "$2 \cdot \mathbb{P}(\Pi)$" will be our reference for the initial depth of terms; while the last section "$3 \cdot \mathbb{P}(\Pi)$" is the final test to see if terms are still growing under recursion, which indicates that a possible propagation of infinite terms can take place. From the size of the number $\mathbb{P}(\Pi)$ that such recursions would have been considered from $2 \cdot \mathbb{P}(\Pi)$ to $3 \cdot \mathbb{P}(\Pi)$.

Now let us denote $\text{POLYLA}(\Pi) = \bigcup_{p \in \text{PRED}(\Pi)} \text{POLYLA}_p(\Pi)$ as the set of all POLY-*limited arguments* of $\text{ARG}(\Pi)$, then we have the following result.

Proposition 7. ***[Containing restricted arguments]*** *Given a program* Π, $\text{AR}(\Pi) \subseteq \text{POLYLA}(\Pi)$.

Definition 5. ***[poly-bounded programs]*** *Given a program* Π, *we say that* Π *is* polynomially bounded, *or simply called* POLY-bounded, *iff* $\text{POLYLA}(\Pi) = \text{ARG}(\Pi)$. *We also denote by* POLY-*bounded as the class of all the* POLY-*bounded programs.*

Intuitively, Definition 5 says that if a program is POLY-bounded, then we have that all arguments cannot grow nor increase in ground-size beyond the number of $2 \cdot \mathbb{P}(\Pi)$ iterations of $\mathcal{U}^k(\Pi, S)$, where $S = \mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$. Note that Definition 5 defines the polynomial bound $2 \cdot \mathbb{P}(\Pi)$ for computing $\mathcal{U}^k(\Pi, S)$, instead of $\mathbb{P}(\Pi)$. This is due to the possibility that the growth of term depth in Π may run through multiple arguments, from which we may only gain sufficient information to predict if the iteration will continue or stop, by computing the second run of iterations through all arguments.

Let Π be the program with just the two single rule (note that $\Pi^{\text{DEF}} = \Pi$):

$$\begin{aligned} \mathsf{r}_1: \ & \mathsf{p}(X, X, X, X, X) \leftarrow \mathsf{b}(X, X, X, X, X), \\ \mathsf{r}_2: \ & \mathsf{p}\big(Y, Z, W, f(V), X\big) \leftarrow \mathsf{p}(X, Y, Z, W, V), \end{aligned}$$

where the second rule propagates all arguments one position to the right where $\mathsf{p}[4]$ contains a function in the head. Then all arguments $\mathsf{p}[i]$ $(1 \leq i \leq 5)$ are in a path of a growing cycle in the argumentation graph $\mathcal{G}_L(\Pi)$. Then with the sequence of SCC: $\langle \{\mathsf{r}_1\}, \{\mathsf{r}_2\} \rangle$, it forms the SCC stratification of $\text{SCC}(\Pi^{\text{DEF}}) = \big\{\{\mathsf{r}_1\}, \{\mathsf{r}_2\}\big\}$ (see Section 2.7). Therefore, for steps $0 \leq i \leq 12$ of $\mathcal{U}^i(\Pi, \emptyset)$, the following sequence of atoms will be generated:

$$
\begin{aligned}
&\mathsf{p}\big(f(V), f(V), f(V), f(V), f(V)\big) && \in \mathcal{U}^0(\Pi, \emptyset),\\
&\mathsf{p}\big(f(V), f(V), f(V), f^2(V), f(V)\big) && \in \mathcal{U}^1(\Pi, \emptyset),\\
&\mathsf{p}\big(f(V), f(V), f^2(V), f^2(V)\big), f(V)\big) && \in \mathcal{U}^2(\Pi, \emptyset),\\
&\mathsf{p}\big(f(V), f^2(V), f^2(V), f^2(V), f(V)\big) && \in \mathcal{U}^3(\Pi, \emptyset),\\
&\mathsf{p}\big(f^2(V), f^2(V), f^2(V), f^2(V), f(V)\big) && \in \mathcal{U}^4(\Pi, \emptyset),\\
&\mathsf{p}\big(f^2(V), f^2(V), f^2(V), f^2(V), f^2(V)\big) && \in \mathcal{U}^5(\Pi, \emptyset),\\
&\mathsf{p}\big(f^2(V), f^2(V), f^2(V), f^3(V), f^2(V)\big) && \in \mathcal{U}^6(\Pi, \emptyset),\\
&\qquad\vdots\ \vdots\ \vdots && \\
&\mathsf{p}\big(f^3(V), f^3(V), f^3(V), f^4(V), f^3(V)\big) && \in \mathcal{U}^{12}(\Pi, \emptyset).
\end{aligned}
$$

In particular, we note that in the second step $\big($i.e., $\mathcal{U}^1(\Pi, \emptyset)$, it only uses the *recursive* rule in the SCC $\{\mathsf{r}_2\}\big)$, so we get from (21) of Definition 3: $\mathcal{U}^1(\Pi, \emptyset) = \Big\{ Hd(\mathsf{r}_2)\theta = \mathsf{p}\big(Y, Z, W, f(V), X\big)\theta \;\Big|\; \theta : \{V, W, X, Y, Z\} \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}}) \cup \mathcal{U}^0(\Pi, \emptyset)\big)\Big\}$. Then with $k = 6$, we have that $\textsc{dep}_{p[4]}\ \big(\mathcal{U}^k(\Pi, \emptyset)\big) < \textsc{dep}_{p[4]}\ \big(\mathcal{U}^{2 \cdot k+1}(\Pi, \emptyset)\big)$. In general, the number $\mathbb{P}(\Pi)$ is the upper-bound for the number K through which we may be able to detect such recursive information as well as the bound on detecting the failing terms growth of restricted arguments.

Example 6. Consider again the program $\Pi_1 = \{\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3\}$ we discussed in Section 1 and its definite normal form $\Pi_1^{\textsc{def}} = \{\mathsf{r}_1, \mathsf{r}'_2, \mathsf{r}''_2, \mathsf{r}_3\}$ given as follows (note that the disjunctive rule $\mathsf{r}_2 \in \Pi_1$ is transformed into the two definite rules $\mathsf{r}'_2, \mathsf{r}''_2 \in \Pi_1^{\textsc{def}}$):

$$
\begin{aligned}
&\mathsf{r}_1 :\ \mathsf{imageViewed}\big(next(X), Y\big) \leftarrow \mathsf{guestFirstViewed}(X, Y), \mathsf{guestMember}(Y),\\
&\mathsf{r}'_2 :\ \mathsf{imageViewed}\big(next(X), Y\big) \leftarrow \mathsf{imageViewed}(X, Y), \mathsf{not}\ \mathsf{skip}\big(next(X), Y\big),\\
&\mathsf{r}''_2 :\ \mathsf{skip}\big(next(X), Y\big) \leftarrow \mathsf{imageViewed}(X, Y), \mathsf{not}\ \mathsf{imageViewed}\big(next(X), Y\big),\\
&\mathsf{r}_3 : \bot \leftarrow \mathsf{imageViewed}\big(next\big(next\big(next\big(next(X)\big)\big)\big), Y\big).
\end{aligned}
$$

By (14) of Definition 2 and the constraint r_4, the initial step $\mathcal{L}^0(\Pi_1)$ will be obtained as follows:

$$
\begin{aligned}
&\mathcal{L}^0(\Pi_1)\\
&= \Big\{\big\langle \mathsf{imageViewed}\big(next\big(next\big(next\big(next(X)\big)\big)\big),Y\big)\theta, -\big\rangle \;\Big|\\
&\qquad\qquad\qquad\qquad\qquad\qquad \theta : \{X,Y\} \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}})\big)\Big\}.\\
&= \Big\{\big\langle \mathsf{imageViewed}\big(next\big(next\big(next\big(next(X)\big)\big)\big),Y\big), -\big\rangle \;\Big|\; X,Y \in \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}})\big)\Big\},\\
&= \Big\{\big\langle \mathsf{imageViewed}\big(next^4(X),Y\big), -\big\rangle \;\Big|\; X,Y \in \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}})\big)\Big\},\\
&= \Big\{\big\langle \mathsf{imageViewed}\big(next^4(Y),X\big), -\big\rangle, \big\langle \mathsf{imageViewed}\big(next^4(Y),next^{i+1}(X)\big), -\big\rangle,\\
&\quad \big\langle \mathsf{imageViewed}\big(next^4(Y),Y\big), -\big\rangle, \big\langle \mathsf{imageViewed}\big(next^4(X),X\big), -\big\rangle\\
&\quad \big\langle \mathsf{imageViewed}\big(next^{4+i}(X),X\big), -\big\rangle, \big\langle \mathsf{imageViewed}\big(next^{4+i}(X),Y\big), -\big\rangle,\\
&\quad \big\langle \mathsf{imageViewed}\big(next^{4+i}(X),next^{i+1}(X)\big), -\big\rangle \;\Big|\; i \in \{1,2,3\}\Big\}, \qquad (37)
\end{aligned}
$$

i.e., we evaluate the atom "$\mathsf{imageViewed}\big(next\big(next\big(next\big(next(X)\big)\big)\big),Y\big)$" for all the X, Y $\in \textsc{terms}\big(\textsc{atoms}(\Pi_1)\big)$, and where we know those atoms are certainly false by the constraint r_4. Then since we cannot infer further certainly true or false atoms from the step $\mathcal{L}^0(\Pi_1)$ of the lower bound, we have that $\mathcal{L}^0(\Pi_1) = \mathcal{L}^1(\Pi_1) = \mathcal{L}^\infty(\Pi_1)$.

Now we compute the upper bound $\mathcal{U}^i(\Pi_1, S)$, where $S = \mathcal{L}^0(\Pi_1)$, and such that $\mathcal{L}^0(\Pi_1)$ $= \mathcal{L}^\infty(\Pi_1)$ as computed above. Firstly, we note that $\textsc{scc}(\Pi_1^{\textsc{def}})^{[0]} = \big\{\{\mathsf{r}_1\}\big\}$, $\textsc{scc}(\Pi_1^{\textsc{def}})^{[1]} = \big\{\{\mathsf{r}_2'\}, \{\mathsf{r}_2''\}, \{\mathsf{r}_3\}, \{\mathsf{r}_4\}\big\}$, corresponds to the SCC stratification of $\textsc{scc}(\Pi_1^{\textsc{def}})$, and such that $\{\mathsf{r}_1\}$, $\{\mathsf{r}_2'\}$, $\{\mathsf{r}_2''\}$, $\{\mathsf{r}_3\}$ and $\{\mathsf{r}_4\}$ are the SCCs of $\Omega(\Pi_1^{\textsc{def}})$ (see Sections 2.6 and 2.7). Then from (20) of Definition 3 we have:

$$
\begin{aligned}
\mathcal{U}^0(\Pi_1, S) &= \Big\{Hd(\mathsf{r}_1)\theta \;\Big|\; \theta : \textsc{var}(\mathsf{r}_1) \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}})\big)\Big\}\\
&= \Big\{\mathsf{imageViewed}\big(next(X),Y\big)\theta \;\Big|\; \theta : \{X,Y\} \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}})\big)\Big\}\\
&= \Big\{\mathsf{imageViewed}\big(next(X),Y\big) \;\Big|\; X,Y \in \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}})\big)\Big\}.
\end{aligned}
$$

On the other hand, from (a) (21) of Definition 3, (b) the only positive body atoms of r_2' and r_2'' are $recrPos(\mathsf{r}_2') = \big\{\mathsf{imageViewed}(X,Y)\big\} = Pos(\mathsf{r}_2')$ and $domPos(\mathsf{r}_2'') = \big\{\mathsf{imageViewed}$ $(X,Y)\big\} = Pos(\mathsf{r}_2'')$, respectively, and (c) r_3 and r_4 are constraints (which will not play any

role in generating atoms in the upper bound), we have that:

$$\begin{aligned}
\mathcal{U}^1(\Pi_1, S) \;=\; & \mathcal{U}^0(\Pi_1, S) \cup \\
& \Big\{ Hd(\mathsf{r}_2')\theta \;\Big|\; \theta : \textsc{var}(\mathsf{r}_2') \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}}) \cup \mathcal{U}^0(\Pi_1, S)\big), \\
& \qquad\qquad Pos(\mathsf{r}_2') \cap S^- \;=\; \emptyset \Big\} \cup \\
& \Big\{ Hd(\mathsf{r}_2'')\theta \;\Big|\; \theta : \textsc{var}(\mathsf{r}_2'') \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}}) \cup \mathcal{U}^0(\Pi_1, S)\big) \text{ and} \\
& \qquad\qquad domPos(\mathsf{r}_2'')\theta \subseteq \mathcal{U}^0(\Pi_1, S)\big) \text{ and } Pos(\mathsf{r}_2'') \cap S^- \;=\; \emptyset \Big\} \\
= \; & \mathcal{U}^0(\Pi_1, S) \cup \\
& \Big\{ \mathsf{imageViewed}\big(next(X), Y\big)\theta \;\Big|\; \theta : \textsc{var}(\mathsf{r}_2') \longrightarrow \\
& \qquad\qquad \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}}) \cup \mathcal{U}^0(\Pi_1, S)\big), Pos(\mathsf{r}_2') \cap S^- \;=\; \emptyset \Big\} \cup \\
& \Big\{ \mathsf{skip}\big(next(X)\big)\theta \;\Big|\; \theta : \textsc{var}(\mathsf{r}_2'') \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}}) \cup \mathcal{U}^0(\Pi_1, S)\big) \text{ and} \\
& \qquad\qquad \big\{\mathsf{imageViewed}(X, Y)\theta\big\} \subseteq \mathcal{U}^0(\Pi_1, S)\big) \text{ and } Pos(\mathsf{r}_2'') \cap S^- \;=\; \emptyset \Big\}.
\end{aligned}$$

Now further from (22) of Definition 3, we get the remaining steps $\mathcal{U}^i(\Pi_1, S)$ (for $i > 1$) as follows:

$$\begin{aligned}
& \mathcal{U}^i(\Pi_1, S) \;=\; \mathcal{U}^{i-1}(\Pi_1, S) \cup \\
& \Big\{ \mathsf{imageViewed}\big(next(X), Y\big)\theta \;\Big|\; \theta : \textsc{var}(\mathsf{r}_2') \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}}) \cup \mathcal{U}^{i-1}(\Pi_1, S)\big), \\
& \qquad\qquad \big\{\mathsf{imageViewed}(X, Y)\theta\big\} \subseteq \mathcal{U}^{i-1}(\Pi_1, S)\big) \text{ and } Pos(\mathsf{r}_2') \cap S^- \;=\; \emptyset \Big\} \cup \qquad (38) \\
& \Big\{ \mathsf{skip}\big(next(X)\big)\theta \;\Big|\; \theta : \textsc{var}(\mathsf{r}_2'') \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi_1^{\textsc{def}}) \cup \mathcal{U}^{i-1}(\Pi_1, S)\big) \text{ and} \\
& \qquad\qquad \big\{\mathsf{imageViewed}(X, Y)\theta\big\} \subseteq \mathcal{U}^1(\Pi_1, S)\big) \text{ and } Pos(\mathsf{r}_2'') \cap S^- \;=\; \emptyset \Big\}. \qquad (39)
\end{aligned}$$

Because we require $domPos(\mathsf{r}_2'')\theta = \big\{\mathsf{imageViewed}(X, Y)\theta\big\} \subseteq \mathcal{U}^1(\Pi_1, S)$ in (39), i.e., $domPos(\mathsf{r}_2'')\theta$ to be confined only to the atoms derived from the step $\mathcal{U}^1(\Pi_1, S)$, it follows that it is only through the set (38) above to generate a possibly infinite growth of complex terms via the recursive application of rule r_2'. On the other hand, because of the fact that S^- contains the set of atoms given by the set (37), it follows through the condition $Pos(\mathsf{r}_2') = \big\{\mathsf{imageViewed}(X, Y)\theta\big\} \cap S^- = \emptyset$ of (38) that the depth of the terms as derived in the atom $Hd(\mathsf{r}_2'')\theta$ is bounded by the lower bound $\mathcal{L}^0(\Pi_1) = S$. Therefore, it follows that program Π_1 is poly-bounded.

Now let $D = \big\{\mathsf{guestFirstViewed}(panda, john) \leftarrow \top,\ \mathsf{guestMember}(john) \leftarrow \top\big\}$ be an input database. Then it follows that $\Pi_1 \cup D$ will only have a finite set of stable models with the following form:

$$M_k \;=\; D \cup \big\{\mathsf{imageViewed}\big(next^i(panda), john\big), \mathsf{skip}\big(next^{i+1}(panda)\big) \;\big|\; i \leq k\big\},$$

where $k \in \{1, 2, 3\}$. □

Theorem 2. ***[Finite groundness]*** *If* Π *is* POLY*-bounded, then for every input database* D *(*D *can be empty), program* $\Pi \cup D$ *is finitely ground.*

Example 7. Consider the program $\Pi[k]$ of Example 4 once again. Because in the steps (34) (where $i > k$) we restrict $domPos(r)\theta$ of those rules r considered in (34) to be the only one of the previous step k, it follows that $\text{DEP}_{p[i]}\big(\mathcal{U}^{2\cdot\mathbb{P}(\Pi[k])}\big(\Pi[k], S\big)\big) = \text{DEP}_{p[i]}\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi[k])}\big(\Pi[k], S\big)\big)$, for all $p[i] \in \text{ARG}\big(\Pi[k]\big)$. Thus, program $\Pi[k]$ is POLY-bounded. □

As we have seen from Example 7 above, the depth of the terms mentioned in any atom in $\mathcal{U}^{\mathbb{P}(\Pi[k])}\big(\Pi[k], S\big)$ is bounded by the polynomial $\mathbb{P}(\Pi[k])$. The next result shows that a stable model of the program $\Pi[k]$ can actually have a term depth that is exponential to the size of program.

Proposition 8. *Let* $D = \{p(a, a, a)\}$*, where* $a \in \mathsf{Const}$ *is a constant, be a database and* $\Pi[k]$ *(*$k > 1$*) be the program as defined in Example 4. Then we have that* $\text{DEP}_{p[i]}(M) \geq O(2^{\mathbb{P}(\Pi[k])})$ *for any stable model* M *of* $\Pi[k]$*.*

Proof. First, we note that $\Pi[k]$ is a positive normal program, so it follows that $M = \mathcal{L}^{\infty}\big(\Pi[k] \cup D\big)^{+}$ is actually its unique stable model. Now we know that through iterative applications of rules (27) and (28), the term depth of $p_k[1]$ is doubled from that of $p_{(k-1)}[1]$ obtained from the previous program $\Pi[k-1]$, while the last rule (29) simply makes $p_{k+1}(Y, X, X)$ be ready for the next program $\Pi[k+1]$. It can be shown that for each $k > 1$, we have

$$\text{DEP}_{p_k[1]}\big(\mathcal{L}^{k}\big(\Pi[k] \cup D\big)\big) = 2^k.$$

Now because $\big|\Pi[k]^{\text{DEF}}\big| = 2 + 3k$ and $\text{MAXART}\big(\Pi[k]\big) \cdot \text{MAXPOS}\big(\Pi[k]\big) = 3$ for all $k > 1$, we have from (35) that $\mathbb{P}\big(\Pi[k]\big) = 3(2+3k) + \big[3(2+3k)\big]^3$. Let $l(k) = \mathbb{P}\big(\Pi[k]\big)$, then it follows that

$$\text{DEP}_{p_{(2l(k))}[1]}\big(\mathcal{L}^{2l(k)+1}\big(\Pi[2l(k)]\big)\big) \;<\; \text{DEP}_{p_{3l(k)}[1]}\big(\mathcal{L}^{3l(k)}\big(\Pi[3l(k)]\big)\big),$$

holds for all $k > 1$. □

4.1 Comparison with Other Decidable Classes of Programs with Function Symbols

In this section, we compare our new POLY-bounded class with the two classes GMT-bounded (Greco et al., 2013) and SIZE-restricted (Calautti et al., 2015a) since they currently contain all the decidable classes of logic programs with function symbols. Thus, it is sufficient to show that our POLY-bounded class strictly contains these two aforementioned classes in order to show that our new class now forms the largest decidable class of programs with function symbols. For reference, we provide overviews of the GMT-bounded and SIZE-restricted classes of programs in Appendixes A.2 and A.1, respectively.

The following two theorems (Theorems 3 and 4) simply indicate that our POLY-bounded programs form the largest decidable class of finitely ground programs, that are discovered in literature so far. We first present a result about the complexity upper bound of deciding GMT-bounded and SIZE-restricted classes.

Proposition 9. *[**gmt-bounded and size-restricted membership complexity**] Deciding whether a program* Π *is* GMT*-bounded or* SIZE*-restricted is in* PSPACE*.*

Proof. We first show that deciding whether Π is GMT-bounded is in PSPACE. Indeed, since enumerating all basic cycles in a directed graph is space bounded by $O(n + e)$, where n is the number of nodes and e is the number of edges (Johnson, 1975), and since the size of the labelled argumentation graph and each cycle length is polynomial in the size of the program Π, it follows from Definition 8 (see Appendix A) that deciding GMT-boundedness is in PSPACE. On the other hand, from Calautti, Greco, Molinaro and Trubitsyna (2015a), we know that for a given set of limited arguments $\mathcal{A}$, deciding whether Π is SIZE-restricted under the arguments in $\mathcal{A}$ is in NP. This follows that when $\mathcal{A}$ is the GMT-limited , deciding whether Π is SIZE-restricted is also in PSPACE. □

Theorem 3. *[**Strictly contains gmt-bounded**] The* POLY*-bounded class strictly contains the* GMT*-bounded class.*

Theorem 4. *[**Strictly contains size-restricted**] The* POLY*-bounded class strictly contains the* SIZE*-restricted class.*

Example 8. Let Π be a program consisting of the following rules:

$$\begin{aligned}
&\mathsf{r}_1: \ \mathsf{q}\big(h(X), Y\big) \leftarrow \mathsf{p}_1(X, Y),\\
&\mathsf{r}_2: \ \mathsf{r}\big(X, g(Y)\big) \leftarrow \mathsf{p}_2(X, Y),\\
&\mathsf{r}_3: \ \mathsf{s}(X, Z) \leftarrow \mathsf{q}(X, Y),\ \mathsf{r}(Y, Z),\\
&\mathsf{r}_4: \ \mathsf{t}\big(f(X)\big) \leftarrow \mathsf{t}(X),\ \mathsf{s}\big(g(Y), h(Z)\big),
\end{aligned}$$

where p_1 and p_2 are extensional predicate symbols. Clearly, we have that $\Pi^{\text{DEF}} = \Pi$ and $\text{SCC}(\Pi^{\text{DEF}}) = \big\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4\big\}$, where $\mathcal{C}_1 = \{\mathsf{r}_1\}$, $\mathcal{C}_2 = \{\mathsf{r}_2\}$, $\mathcal{C}_3 = \{\mathsf{r}_3\}$, and $\mathcal{C}_4 = \{\mathsf{r}_4\}$, which correspond to the SCC of the firing graph $\Omega(\Pi^{\text{DEF}})$[8].

Moreover, for each rule r_i ($i \in \{1, 2, 3, 4\}$), we have (1) $extPos(\mathsf{r}_1) = \big\{\mathsf{p}_1(X)\big\}$, $domPos(\mathsf{r}_1) = \emptyset$, $recrPos(\mathsf{r}_1) = \emptyset$; (2) $extPos(\mathsf{r}_2) = \big\{\mathsf{p}_2(X)\big\}$, $domPos(\mathsf{r}_1) = \emptyset$, $recrPos(\mathsf{r}_1) = \emptyset$; (3) $extPos(\mathsf{r}_3) = \emptyset$, $domPos(\mathsf{r}_3) = \big\{\mathsf{q}(X, Y), \mathsf{r}(Y, Z)\big\}$, $recrPos(\mathsf{r}_3) = \emptyset$; and (4) $extPos(\mathsf{r}_4) = \emptyset$, $domPos(\mathsf{r}_4) = \big\{\mathsf{s}\big(g(Y), h(Z)\big)\big\}$, $recrPos(\mathsf{r}_4) = \big\{\mathsf{t}(X)\big\}$.

Then we obtain the SCC sequence $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$, $\text{SCC}(\Pi^{\text{DEF}})^{[1]}$, $\text{SCC}(\Pi^{\text{DEF}})^{[2]}$, where $\text{SCC}(\Pi^{\text{DEF}})^{[0]} = \{\mathcal{C}_1, \mathcal{C}_2\}$, $\text{SCC}(\Pi^{\text{DEF}})^{[1]} = \{\mathcal{C}_3\}$ and $\text{SCC}(\Pi^{\text{DEF}})^{[2]} = \{\mathcal{C}_4\}$, which is the SCC stratification sequence of $\text{SCC}(\Pi^{\text{DEF}})$ (see Section 2.7). Then according to Definition 3, we can compute the upper bound $\mathcal{U}^i(\Pi, \emptyset)$ for the three subcases: $= 0$, $i = 1, 2$, and $i > 2$. Here we set the input $S \subseteq \mathcal{L}^\infty(\Pi)$ to be empty, i.e., $S = \emptyset$).

For the initial step $i = 0$:

$$\begin{aligned}
\mathcal{U}^0(\Pi, \emptyset) \ = \ & \big\{Hd(\mathsf{r}_1)\theta \ \mid \ \theta : \text{VAR}(\mathsf{r}_1) \longrightarrow \text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}})\big)\big\} \ \cup\\
& \big\{Hd(\mathsf{r}_2)\theta \ \mid \ \theta : \text{VAR}(\mathsf{r}_2) \longrightarrow \text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}})\big)\big\}\\
= \ & \Big\{\mathsf{q}\big(h(X), Y\big)\theta \ \mid \ \theta : \text{VAR}(\mathsf{r}_1) \longrightarrow \big\{X,\, Y,\, Z,\, h(X),\, h(Z),\, g(Y),\, f(X)\big\}\Big\} \ \cup\\
& \Big\{\mathsf{r}\big(X, g(Y)\big)\theta \ \mid \ \theta : \text{VAR}(\mathsf{r}_2) \longrightarrow \big\{X,\, Y,\, Z,\, h(X),\, h(Z),\, g(Y),\, f(X)\big\}\Big\},
\end{aligned}$$

8. See Sections 2.6 and 2.7 for the notions of the firing graph $\Omega(\Pi^{\text{DEF}})$ and SCC $\text{SCC}(\Pi^{\text{DEF}})$.

where $\textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}})\big) = \big\{X, Y, Z, h(X), h(Z), g(Y), f(X)\big\}$, i.e., all the terms and subterms mentioned in $\textsc{atoms}(\Pi^{\textsc{def}})$.

Then from (21) of Definition 3, we have the following for $i = 1$ and $i = 2$, respectively:

$$\begin{aligned}
\mathcal{U}^1(\Pi, \emptyset) &= \mathcal{U}^0(\Pi, \emptyset) \cup \big\{Hd(\mathsf{r}_3)\theta \mid \theta : \textsc{var}(\mathsf{r}_3) \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}}) \cup \mathcal{U}^0(\Pi, \emptyset)\big) \text{ and} \\
&\qquad\qquad domPos(\mathsf{r}_3)\theta \subseteq \mathcal{U}^0(\Pi, \emptyset)\big\} \\
&= \mathcal{U}^0(\Pi, \emptyset) \cup \Big\{\mathsf{s}(X, Z)\theta \mid \theta : \textsc{var}(\mathsf{r}_3) \longrightarrow \big\{X, Y, Z, h(X), h(Z), g(Y), f(X)\big\} \cup \\
&\qquad\qquad \big\{h(X), h(Y), h(Z), h\big(h(X)\big), h\big(h(Z)\big), h\big(g(Y)\big), h\big(f(X)\big)\big\} \cup \\
&\qquad\qquad \big\{g(X), g(Y), g(Z), g\big(h(X)\big), g\big(h(Z)\big), g\big(g(Y)\big), g\big(f(X)\big)\big\} \\
&\qquad\qquad \text{and } domPos(\mathsf{r}_3) = \big\{\mathsf{q}(X, Y)\theta, \mathsf{r}(Y, Z)\theta\big\} \subseteq \mathcal{U}^0(\Pi, \emptyset)\Big\} \\
&= \mathcal{U}^0(\Pi, \emptyset) \cup \Big\{\mathsf{s}\big(h\big(\theta(X)\big), g\big(\theta(Y)\big)\big) \mid \\
&\qquad\qquad \theta : \{X, Y\} \longrightarrow \big\{X, Y, Z, h(X), h(Z), g(Y), f(X)\big\}\Big\},
\end{aligned}$$

$$\begin{aligned}
\mathcal{U}^2(\Pi, \emptyset) &= \mathcal{U}^1(\Pi, \emptyset) \cup \big\{Hd(\mathsf{r}_4)\theta \mid \theta : \textsc{var}(\mathsf{r}_4) \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}}) \cup \mathcal{U}^1(\Pi, \emptyset)\big) \text{ and} \\
&\qquad\qquad domPos(\mathsf{r}_4)\theta \subseteq \mathcal{U}^1(\Pi, \emptyset)\big\} \\
&= \mathcal{U}^1(\Pi, \emptyset) \cup \Big\{\mathsf{t}\big(f(X)\big)\theta \mid \theta : \textsc{var}(\mathsf{r}_4) \longrightarrow \big\{X, Y, Z, h(X), h(Z), g(Y), f(X)\big\} \\
&\qquad\qquad \cup \big\{h(X), h(Y), h(Z), h\big(h(X)\big), h\big(h(Z)\big), h\big(g(Y)\big), h\big(f(X)\big)\big\} \\
&\qquad\qquad \cup \big\{g(X), g(Y), g(Z), g\big(h(X)\big), g\big(h(Z)\big), g\big(g(Y)\big), g\big(f(X)\big)\big\} \\
&\qquad\qquad \text{and } domPos(\mathsf{r}_4) = \big\{\mathsf{s}\big(g(Y), h(Z)\big)\theta\big\} \subseteq \mathcal{U}^1(\Pi, \emptyset)\Big\},
\end{aligned} \tag{40}$$

Finally, based on previous steps, from (22) of Definition 3, we have $\mathcal{U}^3(\Pi, \emptyset)$:

$$\begin{aligned}
\mathcal{U}^3(\Pi, \emptyset) = \mathcal{U}^2(\Pi, \emptyset) \cup \big\{Hd(\mathsf{r})\theta \mid \theta : \textsc{var}(\mathsf{r}) \longrightarrow \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}}) \cup \mathcal{U}^2(\Pi, \emptyset)\big) \text{ and} \\
(1)\ \textsc{terms}\big(extPos(\mathsf{r})\theta\big) \subseteq \textsc{terms}\big(\textsc{atoms}(\Pi^{\textsc{def}})\big); \\
(2)\ domPos(\mathsf{r})\theta \subseteq \mathcal{U}^2(\Pi, \emptyset); \\
(3)\ recrPos(r)(\mathsf{r})\theta \subseteq \mathcal{U}^2(\Pi, \emptyset)\big\}.
\end{aligned}$$

Since rule r_4 is the only rule where $recrPos(\mathsf{r}_4) = \big\{\mathsf{t}(X)\big\} \neq \emptyset$, it is clear that any new atoms that will be added in $\mathcal{U}^3(\Pi, \emptyset), \mathcal{U}^4(\Pi, \emptyset), \mathcal{U}^5(\Pi, \emptyset)$, etc., will be through the head atom $\mathsf{t}\big(f(X)\big)$ of r_4. On the other hand, from $\mathcal{U}^2(\Pi, \emptyset)$, we observe that rule r_4 will not actually be trigged under any assignment because its domain atom $\mathsf{s}\big(g(Y), h(Z)\big) \in domPos(\mathsf{r}_4)$ is not unifiable with any atoms of $\mathcal{U}^1(\Pi, \emptyset)$. Hence, then we can never initiate the recursive application of rule r_4 will never be proceeded. This follows that $\mathcal{U}^3(\Pi, \emptyset) = \mathcal{U}^2(\Pi, \emptyset) = \mathcal{U}^\infty(\Pi, \emptyset)$. □

From Example 8, we can see that there are cases where the lower bound $\mathcal{L}^{\infty}(\Pi)$ is empty, but the program will still terminate from the computation of the upper bound alone. Also note that program Π in Example 8 is neither GMT-bounded nor SIZE-restricted, but terminates because $\mathcal{U}^3(\Pi,\emptyset) = \mathcal{U}^2(\Pi,\emptyset) = \mathcal{U}^{\infty}(\Pi,\emptyset)$ as we have shown above. We will make this result precise in Proposition 13 of Section 6.

5. Exponentially Bounded Programs

In the previous section, we studied the case when the number of the bound is polynomial in the size of the program. Although imposing such a polynomial bound is already enough to strictly contain all the previous decidable classes of programs with function symbols, as stated in Theorems 3 and 4, we observe that there are some applications where programs terminate in exponential number of steps in the size of the underlying program. So in this section, we further generalize our notion of polynomial bounded programs to a notion of exponentially bounded programs.

For this purpose, we firstly introduce some useful notions. Given a program Π and an integer $k \geq 0$, we define another number $\mathsf{exp}_{\Pi}(k)$ inductively as follows:

(1) $\mathsf{exp}_{\Pi}(0) = \mathbb{P}(\Pi)$, and

(2) $\mathsf{exp}_{\Pi}(k+1) = 2^{\mathsf{exp}_{\Pi}(k)}$.

Intuitively, the number $\mathsf{exp}_{\Pi}(k)$ denotes the k-nested number of the expression:

$$2^{2^{\cdot^{\cdot^{\cdot^{2^{\mathbb{P}(\Pi)}}}}}}.$$

Note that $\mathsf{exp}_{\Pi}(0) = \mathbb{P}(\Pi)$ while $\mathsf{exp}_{\Pi}(1) = 2^{\mathbb{P}(\Pi)}$.

Definition 6. *Given a program Π, a predicate $p \in \text{PRED}(\Pi)$ and an integer $k \geq 0$, we denote by k-ExpLA$_p(\Pi)$ as the following set of arguments:*

$$\Big\{\, p[i] \mid i \in \{1,\ldots,\text{ARITY}(p)\} \text{ and } \text{DEP}_{p[i]}\big(\mathcal{U}^{2\cdot \mathsf{exp}_{\Pi}(k)}(\Pi,S)\big) = \text{DEP}_{p[i]}\big(\mathcal{U}^{3\cdot \mathsf{exp}_{\Pi}(k)}(\Pi,S)\big) \Big\},$$

where $S = \mathcal{L}^{\mathsf{exp}_{\Pi}(k)}(\Pi)$.

Similarly to the case of polyLA$_p(\Pi)$ in Definition 4, k-ExpLA$_p(\Pi)$ denotes the set of arguments of the predicate p that does not increase in depth beyond the iterations $\mathsf{exp}_{\Pi}(k)$ of $\mathcal{U}^k(\Pi,S)$, where S is the set of (not necessarily ground) atoms obtained from $\mathcal{L}^{\mathsf{exp}_{\Pi}(k)}(\Pi)$. More generally, we call the arguments in the set $\bigcup_{p\,\in\,\text{PRED}(\Pi)} k\text{-ExpLA}_p(\Pi)$ as the *k-exp-limited arguments* of ARG(Π), which we denote by k-ExpLA(Π). Clearly, when $k = 0$, we have that 0-ExpLA(Π) = polyLA(Π).

Definition 7. *[k-**exp-bounded programs**] Given a program Π and an integer $k \geq 0$, we say that Π is k-*exponentially bounded*, or simply called k-*exp-bounded*, iff k-ExpLA(Π) = ARG(Π). We also denote by k-exp-bounded as the class of all the k-exp-bounded programs.*

Theorem 5. ***[Finite groundness]*** *Given a program* Π *and an integer* $k \geq 0$*, if* Π *is* k-EXP-bounded*, then for every input database* D *(*D *can be empty), program* $\Pi \cup D$ *is finitely ground.*

Proposition 10. ***[0-exp-bounded is poly-bounded]*** *The class of* 0-EXP*-bounded programs is exactly the* POLY*-bounded class.*

6. Computational Complexity

In this section, we study the complexity properties of membership decision for both POLY-bounded and k-EXP-bounded programs, as well as upper bounds of their respective combined complexities.

Theorem 6. ***[poly-bounded membership complexity]*** *Deciding whether a program* Π *is* POLY*-bounded is* EXPTIME*-complete. The hardness holds even if* Π*'s maximum function arity is 2.*

According to Theorem 6, we can see that comparing to other decidable classes, the membership of POLY-bounded programs requires extra computations, which is consistent with the fact that this class of programs strictly contains all previous decidable classes.

Now, we denote by POLY-bounded-$\emptyset$ as the class of programs that are POLY-bounded, where $S = \emptyset$ in Condition (36) in Definition 4, i.e., programs that are POLY-bounded even if the computation of the lower bound $\mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$ is omitted. Then the following two propositions are a direct consequence of Lemma 7 that is used in the proof of Theorem 6 (see B.7).

Proposition 11. *Deciding whether a program is in* POLY*-bounded-*$\emptyset$ *class is* EXPTIME*-complete.*

Proof. (*Membership*) This follows from Theorem 6 since it is a special case of POLY-bounded where we do not compute the lower bound $\mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$.

(*Hardness*) Hardness follows from the program $\Pi_{M(\mathbf{s})} = \Pi^{\text{ORD}}_{M(\mathbf{s})} \cup \Pi^{\text{STR}}_{M(\mathbf{s})} \cup \Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})} \cup \Pi^{\text{ACCEPT}}_{M(\mathbf{s})} \cup \Pi^{\text{UNBOUND}}_{M(\mathbf{s})}$ as constructed in the hardness proof of Theorem 6 (see B.7) and the result of Lemma 7 within the proof, where we assumed the lower bound to be empty (see B.7 for Lemma 7). □

Proposition 12. *Given a* POLY*-bounded program* Π *and a database* D*, the combined complexity of taking* $\Pi \cup D$ *as input is* 2-NEXPTIME*-hard (lower-bound), and is in* 4-EXPTIME *(upper-bound).*

Proof. (*Lower-bound*) This follows from Theorems 3 and 4 that POLY-bounded class contains both the GMT-bounded and SIZE-restricted classes. On the other hand, from the fact that both classes already contain the ω-restricted class (Greco et al., 2013; Calautti et al., 2015a), which has been proved by Syrjänen (2001) to be 2-NEXPTIME-complete in combined complexity, this concludes that the combined complexity of the POLY-bounded class is at least 2-NEXPTIME-hard.

(*Upper-bound*) Before proceeding the proof, we first introduce the notion of term and atom size. For a given term t, we define $\textsc{termSize}(t)$ as the *size* of term t as follows:

$$\textsc{termSize}(t) = \begin{cases} 1 & \text{if } t \text{ is a constant } c \text{ or a variable } X \\ 1 + \sum_{i=1}^{m} \textsc{termSize}(t_i) & \text{if } t = f(t_1, \ldots, t_m). \end{cases}$$

Intuitively, differently from "grsize" as introduce in Section 2.2 and only for the purpose of this proof, our notion termSize is to define the size of the possible number of different simple terms and function symbols that may occur in all the subterms of t. Similarly, we also define a similar notion for atoms. As such, for a given atom $A = p(t_1, \ldots, t_n)$, we set $\textsc{atomSize}(A) = 1 + \sum_{i=1}^{i=n} \textsc{termSize}(t_i)$. Intuitively, $\textsc{atomSize}(A)$ also increments the size of $\sum_{i=1}^{i=n} \textsc{termSize}(t_i)$ by one because we also want to take the occurrence of the predicate symbol in $A = p(t_1, \ldots, t_n)$ (which is p) as one that can vary place holder for all different predicate symbols in $\textsc{pred}(\Pi)$.

Now we are ready to prove the upper-bound. According to Definition 4 and Lemma 3 as used in the proof of Theorem 2 (see B.3), the fact that Π is poly-bounded sets a bound on the maximum size of terms in each atoms in the upper bound $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)$.

On the other hand, recalling that $\textsc{maxArt}(\Pi)$ denotes the product of both the maximum arity of a relational and function symbol in $\textsc{pred}(\Pi) \cup \textsc{funct}(\Pi)$ while $\textsc{maxDep}(\Pi)$ denotes the maximum depth of a term mentioned in the program $\textsc{pred}(\Pi)$ (see their definitions in the third paragraph of section 4). Then, since each application of rules in the derivation of atoms in $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)$ can utmost increase each term's depth by "$\textsc{maxDep}(\Pi)$" in each new derived atom, and where for the previous term depths, the depth for future atom can increase by $\textsc{maxArt}(\Pi)$-times (see for instance, Proposition 8), then the maximum depth that an atom A can be derived in the program Π through $\mathbb{P}(\Pi)$-times applications of its rules is bounded by

$$\textsc{maxArt}(\Pi)^{\mathbb{P}(\Pi)} \cdot \textsc{maxDep}(\Pi) \leq 2^{\left[\mathbb{P}(\Pi) \cdot \textsc{maxArt}(\Pi) + \textsc{maxDep}(\Pi)\right]} = \textsc{maxAtomDep}(\Pi).$$

Therefore, the maximum size $\textsc{atomSize}(A)$ that an atom A can be derived through the rules of Π is bounded by

$$\textsc{maxArt}(\Pi)^{\textsc{maxAtomDep}(\Pi)} \leq 2^{\left[\textsc{maxArt}(\Pi) \cdot \textsc{maxAtomDep}(\Pi)\right]} = \textsc{maxAtomSize}(\Pi).$$

On the other hand, if we further take the predicate symbols in $\textsc{pred}(\Pi)$, the constants mentioned in $\textsc{const}(\Pi \cup D)$, as well as the function symbols in $\textsc{funct}(\Pi)$ into account, then the possible number of different atoms A of size $\textsc{atomSize}(A) \leq \textsc{maxAtomSize}(\Pi)$ that can be derived from the rules of Π is bounded by

$$\begin{aligned} &\left|\textsc{pred}(\Pi) \cup \textsc{const}(\Pi \cup D) \cup \textsc{funct}(\Pi)\right|^{\textsc{maxAtomSize}(\Pi)} \\ \leq\, &2^{\left[\,|\,\textsc{pred}(\Pi) \cup \textsc{const}(\Pi \cup D) \cup \textsc{funct}(\Pi)\,| \cdot \textsc{maxAtomSize}(\Pi)\right]} = \textsc{maxAtoms}(\Pi). \end{aligned}$$

(i.e., 3-EXP). That is, we can think of the simple terms (i.e., the variables and constants) mentioned in each atom (for which whose size is bounded by "$\textsc{maxAtomSize}(\Pi)$") as "place holders" for the different predicates, constants as well as function symbols mentioned in Π. Then finally, since (at the worst case), we need to consider each of the possible subsets,

whose size is bounded by $2^{\text{MAXATOMS}(\Pi)}$, in the checking for a *stable model* (which we do by performing the *reduct* (see Section 2.4) of the program and then the "fixpoint-type" operation to determine if the atoms consider are well-founded with respect to the reduct), then it follows that the whole process of checking if $\Pi \cup D$ has a stable model by taking both Π and D as input is in 4-EXPTIME. □

Proposition 13. POLY-*bounded-*$\emptyset$ *class strictly contains both the* GMT-*bounded and the* SIZE-*restricted classes.*

Proof. ($\subseteq$) Membership follows from the proofs of Theorems 3 and 4 (see B.4 and B.5, respectively) where we assume, without loss of generality due to Proposition 4, that the lower bound corresponding to set S is $\emptyset$.

($\subsetneq$) The program in Example 8 as well as the program $\Pi_{M(\mathbf{s})} = \Pi^{\text{ORD}}_{M(\mathbf{s})} \cup \Pi^{\text{STR}}_{M(\mathbf{s})} \cup \Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})} \cup \Pi^{\text{ACCEPT}}_{M(\mathbf{s})} \cup \Pi^{\text{UNBOUND}}_{M(\mathbf{s})}$ in the hardness proof of Theorem 6 are neither GMT-bounded nor SIZE-restricted but is in the POLY-bounded-$\emptyset$ class. This completes the proof of Proposition 13. □

Finally we have the following membership complexity result for the classes of k-EXP-bounded logic programs.

Theorem 7. *[*k**-exp-bounded membership complexity]** *For $k \geq 1$, deciding whether a program Π is k-*EXP*-bounded is $(k+1)$-*EXPTIME*-complete.*

7. Case Study: Propositional Planning via ASP with Functions

In this section, we consider the propositional planning problem, from which, we demonstrate how our k-EXP-bounded programs will be useful in the planning domain.

First we present the overview of propositional planning. Following the notions introduced by Bylander (1994), a *propositional planning system* is a tuple $(\mathcal{P}, \mathcal{O}, \mathcal{I}, \mathcal{G})$, where:

- $\mathcal{P} = \{c_1, \ldots, c_{|\mathcal{P}|}\}$ is a finite set of *propositional atoms*, called the *conditions*;
- $\mathcal{O}$ is a finite set of *operators*, where each operator is armed with a tuple $(\delta, \lambda, \kappa, \iota)$:
  - $\delta \subseteq \mathcal{P}$ is a set of *positive preconditions*;
  - $\lambda \subseteq \mathcal{P}$ is a set of *negative preconditions*;
  - $\kappa \subseteq \mathcal{P}$ is a set of *positive postconditions*;
  - $\iota \subseteq \mathcal{P}$ is a set of *negative postconditions*; and
  - $\delta \cap \lambda = \emptyset$ and $\kappa \cap \iota = \emptyset$.
- $\mathcal{I} \subseteq \mathcal{P}$ is the set of *initial states*; and
- $\mathcal{G} = (\mathcal{M}, \mathcal{N})$ is the *goal*:
  - $\mathcal{M} \subseteq \mathcal{P}$ is a set of *positive goals*;
  - $\mathcal{N} \subseteq \mathcal{P}$ is a set of *negative goals*; and
  - $\mathcal{M} \cap \mathcal{N} = \emptyset$.

A *state* in the system is specified by a subset $\mathcal{S} \subseteq \mathcal{P}$, indicating that $p \in \mathcal{P}$ is true in state $\mathcal{S}$ iff $p \in \mathcal{S}$. $\mathcal{I}$ specifies what conditions are true and false in the initial state, i.e., $p \in \mathcal{P}$ is initially true if $p \in \mathcal{I}$ and initially false otherwise. A state $\mathcal{S}$ is called a *goal state* if $\mathcal{M} \subseteq \mathcal{S}$ and $\mathcal{S} \cap \mathcal{N} = \emptyset$. $\mathcal{O}$ is the set of operators that can change one state to another. Given a state $\mathcal{S}$ and an operator $o = (\delta, \lambda, \kappa, \iota)$, we say that o is *legal* at state $\mathcal{S}$ if $\delta \subseteq \mathcal{S}$ and $\lambda \cap \mathcal{S} = \emptyset$, and *illegal* otherwise; the resulting state $o(\mathcal{S})$ obtained from $\mathcal{S}$ by applying the operator o is then defined as follows:

$$o(\mathcal{S}) := \begin{cases} (\mathcal{S} \cup \kappa) \setminus \iota & \text{if } o \text{ is legal at } \mathcal{S}; \\ \mathcal{S} & \text{otherwise.} \end{cases}$$

A finite sequence of operators $(o_1, o_2, \ldots, o_n)$ is called a *solution* if the state

$$o_n(\cdots o_2(o_1(\mathcal{I}))\cdots)$$

is a goal state. A solution is said to be *succinct* if there are no integers i and j such that $1 \leq i < j \leq n$ and $o_i(\cdots o_2(o_1(\mathcal{I}))\cdots) = o_j(\cdots o_2(o_1(\mathcal{I}))\cdots)$.

To encode a planning problem into *answer set programming* (ASP), we consider a language in a Herbrand domain, where

- `initial`, `goal_pos` and `goal_neg` are three unary predicates that encode the sets of initial conditions, positive goals and negative goals, respectively;
- `precond_pos`, `precond_neg`, `postcond_pos` and `postcond_neg` are four binary predicates that encode the sets of positive preconditions, negative preconditions, positive postconditions and negative postconditions, respectively;
- `action` and `stage` are two unary predicates that encode the set of operators and stages involving in an intended solution;
- $\texttt{maxStage}^0$ is a predicate with arity $2 \cdot |\mathcal{P}| + 1$, while atom $\texttt{maxStage}^0(T, \mathbf{X}, \mathbf{Y})$ is for deriving the complex term T under the (successor) function `s` up to the depth $2^{|\mathcal{P}|}$.
- Using similar notions to the "ordering axioms" in the rules of the program $\Pi^{\text{ORD}}_{M(\mathbf{s})}$ in the hardness proof of Theorem 6 (see Appendix B.7), the predicates $\texttt{minNum}(\mathbf{X})$ and $\texttt{maxNum}(\mathbf{Y})$ of arity $|\mathcal{P}|$ denote the minimum and maximum enumerations, respectively, and simply set the maximum depth at $2^{|\mathcal{P}|}$ (see rule (43));
- Also using similar notions to the "ordering axioms" in the rules of the program $\Pi^{\text{ORD}}_{M(\mathbf{s})}$ in the hardness proof of Theorem 6 (see Appendix B.7), the predicates $\leq$ and $\prec$ denote the less-then-equal and the successor relations among the numbering scheme using the $|\mathcal{P}|$-length tuples, respectively.

We further use υ to denote the set of all extensional predicates. Then each propositional planning system can be encoded by a database over υ, where by a database over υ we mean a finite set of ground atoms built from υ.

Let s be an unary Herbrand function symbol representing the immediate successor relation between stages. Now we specify Π_{PLAN} to be the program as follows:

$$\texttt{maxStage}^0(0, \mathbf{X}, \mathbf{X}) \texttt{ :- } \texttt{minNum}(\mathbf{X}). \tag{41}$$
$$\texttt{maxStage}^0(\texttt{s(T)}, \mathbf{X}, \mathbf{Z}) \texttt{ :- } \texttt{maxStage}^0(T, \mathbf{X}, \mathbf{Y}), \mathbf{Y} \prec \mathbf{Z}, \mathbf{X} \leq \mathbf{Z}. \tag{42}$$
$$\texttt{maxStage(s(T))} \texttt{ :- } \texttt{maxStage}^0(\texttt{T}, \mathbf{X}, \mathbf{Y}), \texttt{minNum}(\mathbf{X}), \texttt{maxNum}(\mathbf{Y}). \tag{43}$$
$$\texttt{cond}(c_1) \texttt{ :-}. \tag{44}$$
$$\vdots$$
$$\texttt{cond}(c_{|\mathcal{P}|}) \texttt{ :-}. \tag{45}$$
$$\texttt{stage(0)} \texttt{ :-}. \tag{46}$$
$$\texttt{true(0,P)} \texttt{ :- initial(P)}. \tag{47}$$
$$\texttt{eq(T,T)} \texttt{ :- stage(T)}. \tag{48}$$
$$\texttt{eq(A,A)} \texttt{ :- action(A)}. \tag{49}$$
$$\texttt{do(T,A)} \texttt{ :- action(A),unsuc(T),not not\_do(T,A)}. \tag{50}$$
$$\texttt{not\_do(T,A)} \texttt{ :- action(A),unsuc(T),not do(T,A)}. \tag{51}$$
$$\texttt{:- do(T,A),do(T,B),not eq(A,B)}. \tag{52}$$
$$\texttt{illegal(T,A)} \texttt{ :- stage(T),precond\_pos(A,P),not true(T,P),cond(P)}. \tag{53}$$
$$\texttt{illegal(T,A)} \texttt{ :- stage(T),precond\_neg(A,P),true(T,P),cond(P)}. \tag{54}$$
$$\texttt{true(s(T),P)} \texttt{ :- do(T,A),postcond\_pos(A,P),not illegal(T,A),.}$$
$$\texttt{not maxStage(s(T)),cond(P)}. \tag{55}$$
$$\texttt{false(s(T),P)} \texttt{ :- do(T,A),postcond\_neg(A,P),not illegal(T,A),}$$
$$\texttt{not maxStage(s(T)),cond(P)}. \tag{56}$$
$$\texttt{true(s(T),P)} \texttt{ :- true(T,P),stage(s(T)),not false(s(T),P),.}$$
$$\texttt{not maxStage(s(T))}. \tag{57}$$
$$\texttt{:- true(T,P),false(T,P)}. \tag{58}$$
$$\texttt{stage(s(T))} \texttt{ :- do(T,A),not maxStage(s(T))}. \tag{59}$$
$$\texttt{diff(T,S)} \texttt{ :- true(T,P),stage(S),not true(S,P)}. \tag{60}$$
$$\texttt{diff(T,S)} \texttt{ :- stage(T),not true(T,P),true(S,P)}. \tag{61}$$
$$\texttt{:- stage(T),stage(S),not eq(T,S),not diff(T,S)}. \tag{62}$$
$$\texttt{unsuc(T)} \texttt{ :- stage(T),goal\_pos(P),not true(T,P)}. \tag{63}$$
$$\texttt{unsuc(T)} \texttt{ :- stage(T),goal\_neg(P),true(T,P)}. \tag{64}$$
$$\texttt{done} \texttt{ :- stage(T),not unsuc(T)}. \tag{65}$$
$$\texttt{:- not done}. \tag{66}$$

where: `action`, `initial`, `goal_pos`, `goal_neg`, `precond_pos`, `precond_neg`, `postcond_pos` and `postcond_neg` are extensional (or base) predicates; and rules (44)-(45) encode the possible conditions c_1, ..., $c_{|\mathcal{P}|}$ such that each c_i (for $i \in \{1, \ldots, |\mathcal{P}|\}$) are constants encoding the condition $c_i \in \mathcal{P}$.

Here predicate `maxStage` (as mentioned in the head of rule (43)) encodes the maximum possible stage that can be derived via the ordering relations `minNum`, `maxNum`, $\leq$ and $\prec$, and where these ordering relations can be defined in a similar manner as the program $\Pi^{\text{ORD}}_{M(\mathbf{s})}$ in the hardness proof of Theorem 6 (see Appendix B.7). As such, the presence of the literal "`not maxStage(s(T))`" in the bodies of rules (55), (56), (57) and (59) bounds the recursive buildup of the complex terms through the successor function symbol `s` since the maximum stage will be in the lower bound $\mathcal{L}^{\mathbb{P}(\Pi_{\text{PLAN}})}(\Pi_{\text{PLAN}})$[9].

In particular, the first rule (41) is for deriving the initial enumeration, "0", which is the first argument (i.e., $\texttt{maxStage}^0[1]$) of the predicate "$\texttt{maxStage}^0$". More precisely, we have that the head atom "$\texttt{maxStage}^0(0, \mathbf{X}, \mathbf{X})$" with "$\texttt{minNum}(\mathbf{X})$" in the body of (41), derives "0" with "$\mathbf{X}$" first corresponding to our minimal enumeration. Next, the rule (42) counts each nesting of applications of the successor function "`s(T)`" in terms of the distance between the enumeration values "$\mathbf{X}$" and "$\mathbf{Z}$" so that the number of nesting of the term "`s(T)`" actually corresponds to the (increasing) distance between the enumeration values "$\mathbf{X}$" and "$\mathbf{Z}$". In particular, we note that the application of the function "`s(T)`" in the first term of the predicate "$\texttt{maxStage}^0$" in rule (42) is bounded because of the two body atoms "$\mathbf{Y} \prec \mathbf{Z}$" and "$\mathbf{X} \leq \mathbf{Z}$" of (42) acting as "guards." Then lastly, the rule (43) projects the maximum depth of the terms "`s(T)`" into the predicate "`maxStage`" by considering the term "T" of the body atom "$\texttt{maxStage}^0$" when the distance between the minimum and maximum enumeration (i.e., $\texttt{minNum}(\mathbf{X})$ and $\texttt{maxNum}(\mathbf{Y})$) is at the maximum.

Then we can compute that program Π_{PLAN} terminates after possible 2^n configurations of n states are reached, where $n = |2^{\mathcal{P}}| \leq 2^{\mathbb{P}(\Pi_{\text{PLAN}})} = \mathsf{exp}_{\Pi_{\text{PLAN}}}(1)$, i.e., the size of all possible states. Therefore, it follows that $\mathcal{U}^{\mathsf{exp}_{\Pi_{\text{PLAN}}}(1)}(\Pi_{\text{PLAN}}, S) = \mathcal{U}^{2 \cdot \mathsf{exp}_{\Pi_{\text{PLAN}}}(1)}(\Pi_{\text{PLAN}}, S) = \mathcal{U}^{3 \cdot \mathsf{exp}_{\cdot \Pi_{\text{PLAN}}}(1)}(\Pi_{\text{PLAN}}, S) = \mathcal{U}^{\infty}(\Pi_{\text{PLAN}}, S)$, where $S = \mathcal{L}^{\mathsf{exp}_{\Pi_{\text{PLAN}}}(1)}(\Pi_{\text{PLAN}})$. So Π_{PLAN} is 1-EXP-bounded.

8. Concluding Remarks

Answer set programming with function symbols has demonstrated its useful applications in various situations. For this purpose, discovering finitely ground programs with function symbols has been an important topic in ASP research. On the other hand, in traditional logic programming paradigm, the problem of program termination has also been extensively studied under the *top-down evaluation* approach over the years (Baselice et al., 2009; Bonatti, 2004; Bruynooghe, Codish, Gallagher, Genaim, & Vanhoof, 2007; Genaim & Codish, 2005; Marchiori, 1996; Nguyen, Giesl, Schneider-Kamp, & Schreye, 2007; Nishida & Vidal, 2010; Ohlebusch, 2001; Schneider-Kamp, Giesl, Serebrenik, & Thiemann, 2009; Schneider-Kamp, Giesl, Ströder, Serebrenik, & Thiemann, 2010; Schreye & Decorte, 1994; Serebrenik & Schreye, 2005; Voets & Schreye, 2011). However, as pointed by Greco, Molinaro and Trubitsyna (2013), under the stable model semantics, these methods are generally inapplicable to identify new finitely ground programs.

In this paper, by proposing the stable model polynomial and k-exponential upper bounds for logic programs, we discovered new decidable classes of finitely ground programs, which

9. For clarity and better presentation, we omit the rules of $\Pi^{\text{ORD}}_{M(\mathbf{s})}$ about the ordering relations as mentioned above.

strictly contain all previous existing decidable classes of logic programs. The overall landscape of finitely ground programs is described by Figure 1 illustrated in Section 1.

We should also mention that the decidable classes of finitely ground programs discovered in this paper are of practical values. Section 7 has shown a case how an exponential bounded program has to be used in solving propositional planning problem. Nevertheless, according to our complexity results, checking whether a program is polynomial or exponential bounded is inevitably hard in general.

Acknowledgements

This research is supported in part by National Natural Science Foundation of China under grants U1836204, 61572221 and 61433006.

Appendix A. Overview of GMT-bounded and SIZE-restricted Classes

A.1 GMT-bounded Programs

A *labeled argumentation graph* of a program Π is a graph $\mathcal{G}_L(\Pi) = (V, E)$, where $V = \text{ARG}(\Pi)$, and there is an edge $(q[j], p[i], \langle \alpha, r, h, k \rangle)$[10] in E iff (i) $r \in \Pi$; (ii) $p(t_1, \ldots, t_i, \ldots, t_n)$ is the h-th atom of $Hd(r)$; (iii) $q(u_1, \ldots, u_j, \ldots, u_m)$ is the k-th atom of $Pos(r)$; (iv) α is further defined as follows: (a) $\alpha = \epsilon$ if $u_j = t_i$; (b) $\alpha = f$ if $u_j = X$ and $t_i = f(\ldots, X, \ldots)$; (c) $\alpha = \overline{f}$ if $u_j = f(\ldots, X, \ldots)$ and $t_i = X$.

A *path* ρ in $\mathcal{G}_L(\Pi)$ is a non-empty sequence $(a_1, b_1, \langle \alpha_1, r_1, h_1, k_1 \rangle)$, ..., $(a_s, b_s, \langle \alpha_s, r_s, h_s, k_s \rangle)$ such that $b_i = a_{i+1}$ for $1 \leq i \leq s-1$. If $a_1 = b_s$, then ρ is called a *cyclic path*. Then based on the path ρ, we define: (i) $\lambda_1(\rho) = \alpha_1, \ldots, \alpha_s$ (i.e., the sequence of function symbols); (ii) $\lambda_2(\rho) = r_1, \ldots, r_s$ (i.e., the sequence of rules); and (iii) $\lambda_3(\rho) = \langle r_1, h_1, k_1 \rangle, \ldots, \langle r_s, h_s, k_s \rangle$ (i.e., the sequence of rules with atoms). Given a cycle π of $\mathcal{G}_L(\Pi)$, we denote by $\tau(\pi)$ as the set of all the cyclic paths that can be obtained from π. Thus, given two cycles π_1 and π_2, we denote by $\pi_1 \approx \pi_2$ iff there exist some $\rho_1 \in \tau(\pi_1)$ and $\rho_2 \in \tau(\pi_2)$ such that $\lambda_3(\rho_1) = \lambda_3(\rho_2)$, i.e., π_1 and π_2 go through the same rules and atoms in Π. We say that a cycle π is *basic* if it does not contain occurrences of the same edges (although it may contain the occurrences of the same nodes). A node $p[i]$ *depends on* a node $q[j]$ in $\mathcal{G}_L(\Pi)$ iff there is a path from $q[j]$ to $p[i]$. Finally, we say that a node $p[i]$ *depends on* a cycle π of $\mathcal{G}_L(\Pi)$ iff it depends on a node $q[j]$ appearing in π. Note that nodes appearing on π depends on the cycle π itself. The *activation graph* of Π, denoted $\mathcal{G}_A(\Pi)$, is the graph (V, E) s.t. $V = \Pi$ and there exists an edge $(r_i, r_j) \in E$ iff r_i *activates* r_j (Greco et al., 2013). We say that r_i *activates* r_j iff there exists two ground rules $r_i \in ground(r_i')$ and $r_j \in ground(r_j')$ and a set of ground atoms D such that (1) $D \not\models r_i'$; (2) $D \models r_j'$; and (3) $D \cup Hd(r_j') \models r_j'$.

Given a program Π and its labelled argumentation graph $\mathcal{G}_L(\Pi)$, we define the recursive information on function applications in the rules through the so-called grammars G_Π and G'_Π as the 4-tuples (N, T, R, S) and $(N'T, R', S)$, respectively, where $N = \{S, S_1, S_2\}$ and $N' = \{S\}$ are the sets of non-terminal symbols; $T = \{f \mid f \in \text{FUNCT}(\Pi)\} \cup \{\overline{f} \mid f \in \text{FUNCT}(\Pi)\}$ is the set of terminal symbols; S is the start symbol; and lastly, R is the set of production rules:

10. We assume without loss of generality that the atoms in $Hd(r)$ and $Pos(r)$ of all the rules $r \in \Pi$ have some sort of ordering.

- $S \rightarrow S_1 f_i S_2, \qquad \forall f_i \in \text{FUNCT}(\Pi),$
- $S_1 \rightarrow f_i S_1 \overline{f}_i S_1 \mid \epsilon, \qquad \forall f_i \in \text{FUNCT}(\Pi),$
- $S_2 \rightarrow (S_1 \mid f_i)\, S_2 \mid \epsilon, \qquad \forall f_i \in \text{FUNCT}(\Pi),$

and R' the set of production rules:

- $S \rightarrow f_i S \overline{f}_i S \mid \epsilon, \qquad \forall f_i \in \text{FUNCT}(\Pi).$

Thus, we denote by $\mathcal{L}(G_\Pi)$ and $\mathcal{L}(G'_\Pi)$ as the languages of the set of strings generated by the grammars G_Π and G'_Π, respectively. For example, assuming $\text{FUNCT}(\Pi) = \{f, g\}$, then the three strings: (1) $g\overline{g}f$; (2) fgg; and (3) $g\overline{g}fgg$ are strings of $\mathcal{L}(G_\Pi)$, while the three strings: (1) $f\overline{f}$; (2) $fg\overline{g}\overline{f}$; and (3) $fg\overline{g}\overline{f}g\overline{g}$ are strings of $\mathcal{L}(G'_\Pi)$. Then given a cycle π of $\mathcal{G}_L(\Pi)$, we say that π is:

- *growing* if there is $\rho \in \tau(\pi)$ s.t. $\lambda_1(\rho) \in \mathcal{L}(G_\Pi)$;
- *balanced* if there is $\rho \in \tau(\pi)$ s.t. $\lambda_1(\rho) \in \mathcal{L}(G'_\Pi)$;
- *failing* otherwise.

Intuitively, a *growing* cycle π allows propagation of more complex terms; a *balanced* cycle π denotes no propagation of complex terms; and the cycle π is *failing* otherwise.

Definition 8. *(Greco et al., 2013) Given a set of arguments A, we define $\Psi_{(\text{GMT},\Pi)}(A)$ to be the set of arguments $q[k]$ such that for each basic cycle π that $q[k]$ depends on, at least one of the following conditions holds:*

1. *π is not active or is not growing;*
2. *π contains an edge $\big(s[j], p[i], \langle f, r, l_1, l_2\rangle\big)$ and letting $p(t_1, \ldots, t_n)$ be the l_1-th atom in the head of r, for every variable X in t_i, there is an atom $b(u_1, \ldots, u_m)$ in $Pos(r)$ such that X appears in a term u_h and $b[h] \in A$;*
3. *there is a basic cycle π' in $\mathcal{G}_L(\Pi)$ such that $\pi' \approx \pi$, π' is not balanced, and π only passes through arguments in A.*

By $\Psi^i_{(\text{GMT},\Pi)}(A)$, we further denote the set of arguments inductively defined by:

$$\Psi^0_{(\text{GMT},\Pi)}(A) = \Psi_{(\text{GMT},\Pi)}(A), \text{ and}$$
$$\Psi^{i+1}_{(\text{GMT},\Pi)}(A) = \Psi^{i+1}_{(\text{GMT},\Pi)}\big(\Psi^i_{(\text{GMT},\Pi)}(A)\big).$$

Then by $\Psi^\infty_{(\text{GMT},\Pi)}(A)$, we denote its fixpoint $\bigcup_{i=0}^{\infty} \Psi^i_{(\text{GMT},\Pi)}(A)$.

It is not difficult to verify from Definition 8 that $A_1 \subseteq A_2$ implies $\Psi_{(\text{GMT},\Pi)}(A_1) \subseteq \Psi_{(\text{GMT},\Pi)}(A_2)$ although the relationship $A \subseteq \Psi_{(\text{GMT},\Pi)}(A)$ does not hold in general.

Then a program Π is GMT-*bounded* iff $\Psi^\infty_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big) = \text{ARG}(\Pi)$ (Greco et al., 2013), where $\text{AR}(\Pi)$ denotes the set of *restricted arguments* (or sometimes just *restricted* for convenience) of Π (Lierler & Lifschitz, 2009). Formally, $\text{AR}(\Pi)$ is the set of arguments of Π such that there exists an assignment $\phi : \text{AR}(\Pi) \longrightarrow \mathbb{Z}^+$ so that for all rules $r \in \Pi$, all

atoms $p(t_1, \ldots, t_n) \in Hd(r)$, and all variables X of t_i, if $p[i] \in \text{AR}(\Pi)$, then we have: (1) there is $q(u_1, \ldots, u_m) \in Pos(r)$ with X occurring in u_j and $q[j] \in \text{AR}(\Pi)$, and (2) $\phi\big(p[i]\big) - \phi\big(q[j]\big) \geq \text{DEP}\big(X, t_i\big) - \text{DEP}\big(X, u_j\big)$. Here $\text{DEP}\big(X, t_i\big)$ denotes the maximum term depth of the variable X in term t_i (Greco et al., 2013). Generally speaking, the restricted arguments $\text{AR}(\Pi)$ of Π are the arguments where their edges in the argumentation graph do not form a growing cycle.

For a given set of arguments $A \subseteq \text{ARG}(\Pi)$ and an argument $p[i] \in \text{ARG}(\Pi)$, we say that $p[i]$ is GMT-*limited* under A iff $p[i] \in \Psi^{\infty}_{(\text{GMT},\Pi)}(A)$.

Example 9. Assume Π to be the program in (Greco et al., 2013) consisting of the following rules:

$$\mathsf{count}(lc(a, lc(b, lc(c, nil))), 0), \tag{67}$$
$$\mathsf{f\text{-}count}(X, L, I) \leftarrow \mathsf{count}\big(lc(X, L), I\big), \tag{68}$$
$$\mathsf{count}\big(L, s(I, 1)\big) \leftarrow \mathsf{f\text{-}count}(X, L, I). \tag{69}$$

Then it can be checked that $\Psi^{\infty}_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi')\big) = \text{ARG}(\Pi)$, so Π' is GMT-bounded. Indeed, the arguments $\mathsf{count}[1]$, $\mathsf{f\text{-}count}[1]$ and $\mathsf{f\text{-}count}[2]$ only depends on the failing cycle between the arguments $\mathsf{f\text{-}count}[2]$ and $\mathsf{count}[1]$ of the cycle between the rules (68) and (69). Therefore, we have that $\Psi_{(\text{GMT},\Pi)}(\emptyset) = \big\{\mathsf{count}[1], \mathsf{f\text{-}count}[1], \mathsf{f\text{-}count}[2]\big\}$. Then finally, because the two rules (68) and (69) passes through the arguments in $\Psi_{(\text{GMT},\Pi)}(\emptyset)$ (which is in a failing cycle), then it follows that even though the cycle among the arguments $\mathsf{count}[2]$ and $\mathsf{f\text{-}count}[3]$ is growing, it will still terminate because the two rules activating the cycle (between $\mathsf{count}[2]$ and $\mathsf{f\text{-}count}[3]$) passes through the arguments in $\Psi_{(\text{GMT},\Pi)}(\emptyset) = \big\{\mathsf{count}[1], \mathsf{f\text{-}count}[1], \mathsf{f\text{-}count}[2]\big\}$. Then since this implies that $\Psi_{(\text{GMT},\Pi)}\big(\Psi_{(\text{GMT},\Pi)}(\emptyset)\big) = \big\{\mathsf{count}[2], \mathsf{f\text{-}count}[3]\big\}$ such that $\Psi_{(\text{GMT},\Pi)}\big(\Psi_{(\text{GMT},\Pi)}(\emptyset)\big) \cup \Psi_{(\text{GMT},\Pi)}(\emptyset) = \text{ARG}(\Pi)$, then it follows from Definition 8 that Π is GMT-bounded. □

A.2 SIZE-restricted Programs

Now we introduce another class of programs called size-restricted programs (Calautti et al., 2015a). For a given program Π, each of its rules of the form (1) is replaced by the following k *normal positive rules*:

$$A_i \leftarrow B_1, \ldots, B_l,\ 1 \leq i \leq k,$$

where constraints are omitted, and we denote this positive normal program as $st(\Pi)$. Since the minimal model of $st(\Pi)$ contains every stable model of Π, the finiteness of the minimal model of $st(\Pi)$ implies the finiteness of the stable model of Π (Calautti, Greco, Spezzano, & Trubitsyna, 2015b), and hence, in the rest of this section, the definition of size-restricted program will be based on positive normal programs (Calautti et al., 2015a).

Let Π be a (positive normal) program. By $\Omega(\Pi)$, we denote it as the *firing graph* of Π, which is a directed graph where $\langle r, r' \rangle$ is an edge of $\Omega(\Pi)$ iff there exists two rules (can be the same) r, $r' \in \Pi$ such that $Hd(r)$ and some atom in $Pos(r')$ are *unifiable*. Intuitively, this means that r may cause r' to "fire." Then the *strongly connected component* (SCC) of $\Omega(\Pi)$ is the maximal set of nodes $\mathcal{C}$ where every node of $\mathcal{C}$ always reaches itself. In this case, we also call such $\mathcal{C}$ is a SCC of program Π

For a given program Π, we assume that all distinct SCCs of $\Omega(\Pi)$ are enumerated as $\mathcal{C}_1, \ldots, \mathcal{C}_n$, and as such, we denote by EXT-ARG(Π) as the set $\{p[i/j] \mid \mathcal{C}_j$ is an SCC of Π and $p[i] \in \text{ARG}(\Pi)\}$.

Then the *extended argumentation graph* of a positive normal program Π, denoted as $\Delta(\Pi)$, is a directed graph with nodes EXT-ARG(Π), and there is an edge $(q[j,k], p[i,l])$ iff:

- $k = l$ and there is a rule $r \in \mathcal{C}_k$ such that: (1) $\text{PRED}\big(Hd(r)\big) = p$; (2) there exists some $B \in Pos(r)$ s.t. $\text{PRED}(B) = q$; (3) the i-th term $Hd(r)$ and j-th of B have a common variable; and (4) there is a rule $r' \in \Pi$ s.t. $Hd(r')$ and B unify, or
- $k \neq l$, $p = q$, $i = j$ and there are two rules $r_1 \in \mathcal{C}_k$ and $r_2 \in \mathcal{C}_l$ s.t. $\text{PRED}\big(Hd(r_1)\big) = p$ and (r_1, r_2) is an edge of $\Omega(\Pi)$.

Intuitively, an edge $(q[j,k], p[i,l])$ of $\Delta(\Pi)$ indicates that there can be a derivation of terms from $q[j]$ in component $\mathcal{C}_k$ to $p[i]$ in component $\mathcal{C}_l$. We further say that an extended argument $p[i/l]$ *depends on* an extended argument $q[j/k]$ if there is a path from $q[j/k]$ to $p[i/l]$ in the graph $\Delta(\Pi)$.

For some rule $r \in \Pi$, we say that the head atom is *mutually recursive* with an atom $B \in Pos(r)$ if there is an SCC $\mathcal{C}$ of Π containing r and another rule r' (which can also be r) such that $Hd(r')$ and B unify. We denote by $recrPos(r)$ as the set of mutually recursive atoms in $Pos(r)$ with $Hd(r)$. For a given program Π and a set $\mathcal{A}$ of limited arguments of Π, a rule $r \in \Pi$ is said to be $\mathcal{A}$-*relevant* if $Hd(r)$ contains at least one variable which does not appear in $Pos(r) \setminus recrPos(r)$ and does not appear is some term t_i of some atom $p(t_1, \ldots, t_n) \in Pos(r)$ s.t. $p[i] \in \mathcal{A}$. Intuitively, rules that are not $\mathcal{A}$ relevant will not be considered in the analysis an SCC since they cannot initiate a propagation of infinite terms.

Now we define the notion of a size of a term. The *size* of a term t, denoted $size(t)$, is defined recursively as follows:

$$size(t) = \begin{cases} x & \text{if } t \text{ is a variable } X \\ m + \sum_{i=1}^{m} size(t_i) & \text{if } t = f(t_1, \ldots, t_m), \end{cases}$$

where x is the integer variable corresponding to X. Thus, the *size* of an atom $A = p(t_1, \ldots, t_n)$, denoted as $size(A)$, is the vector $\big(size(t_1), \ldots, size(t_n)\big)$. For example, for atom $A = p(a, X, f(X, g(X, Y))$, we have that $size(A) = (0, x, 2x + y + 4)$.

Now we define the domain of an argument and a predicate as introduced in (Calautti et al., 2015a). Given a program Π and a set arguments $\mathcal{A}$, the *domain* of an argument $p[i] \in \text{ARG}(\Pi)$ w.r.t. $\mathcal{A}$, denoted $\mathbb{D}_{\mathcal{A}}(p[i])$, is $\mathbb{Z}$ if $p[i] \in \mathcal{A}$, and $\mathbb{N}$ otherwise. Thus, the *domain* of a predicate symbol p of arity n is $\mathbb{D}_{\mathcal{A}}(p) = \mathbb{D}_{\mathcal{A}}(p[1]) \times \ldots \times \mathbb{D}_{\mathcal{A}}(p[n])$.

Definition 9. *(Calautti et al., 2015a) Given a program Π and a set $\mathcal{A}$ of limited arguments, let $\mathcal{C}$ be a SCC of $\Omega(\Pi)$ with $\text{PRED}(\mathcal{C}) = \{p_1, \ldots, p_n\}$. We say that an argument $p_i[j]$ of $\mathcal{C}$ is* $\mathcal{A}$-size-restricted *in $\mathcal{C}$ iff:*

1. *For every rule $r \in \mathcal{C}$ such that $Hd(r) = p_i(t_1, \ldots, t_m)$, for every variable X occurring in t_j, there exists a term u_k of body atom $q(u_1, \ldots, u_{m'})$ s.t. X occurs in u_k and $q[k] \in \mathcal{A}$, or*

2. *There exists n vectors* $\overline{v}_h \in \mathbb{D}_{\mathcal{A}}(p_h)$, $1 \leq h \leq n$, *such that for every* $\mathcal{A}$*-relevant rule* $r \in \mathcal{C}$ *there exists an atom* B *in* $Pos(r)$ *such that if* $\text{PRED}\big(Hd(r)\big) = p_k$ *and* $\text{PRED}(B) = p_l$, *then the following holds:*

   (a) *the constraint* $\overline{v}_l \cdot size(B) \geq \overline{v}_k \cdot size\big(Hd(r)\big)$ *is satisfied for every non-negative values of the integer variables in it;*

   (b) $p_k = p_i$ *implies that either* $\overline{v}_i[j] \neq 0$ *or the constraint* $\overline{v}_l \cdot size(B) > \overline{v}_i \cdot size\big(Hd(r)\big)$ *is satisfied for every non-negative values of the integer variables in it.*

Generally speaking, item 2 above imposes a restriction on the rules implying a kind of "failing-cycle" since the nesting of the terms in the head must be less than or equal to that in the body.

Finally, we are ready to define size-restricted programs as introduced by Greco, Molinaro and Trubitsyna (2015a). Let Π be a program and $\mathcal{A}$ a set of limited arguments of Π. Then an argument $p[i]$ is $\mathcal{A}$-size-restricted in Π if for every SCC $\mathcal{C}_l$ of Π where $p \in \text{PRED}(\mathcal{C}_l)$, we have that: (1) $p[i]$ is $\mathcal{A}$-size-restricted in $\mathcal{C}_l$; and (2) $p[i/l]$ depends only on the extended arguments $q[j/k]$ such that $q[j]$ is $\mathcal{A}$-size-restricted in $\mathcal{C}_k$. Then, by $\mathcal{R}_{\mathcal{A}}(\Pi)$, we denote the set of all the $\mathcal{A}$-size-restricted arguments of Π.

Although a direct application of the size-restricted criterion does not directly capture the restricted arguments (Lierler & Lifschitz, 2009) nor the GMT-bounded classes (Greco et al., 2013), it can nevertheless be incorporated by setting the initial set of "limited" arguments $\mathcal{A}$ to be that derived from other bounded classes (Calautti et al., 2015a).

Formally, a program Π is called SIZE-*restricted*, iff $\text{ARG}(\Pi) = \Psi^{\infty}_{(\text{SR},\Pi)}(\mathcal{A} \cup \mathcal{B})$, where $\mathcal{A} = \text{AR}(\Pi)$ and $\mathcal{B}$ are the sets of restricted and GMT-limited arguments of Π, respectively, and the fixpoint operator $\Psi^{\infty}_{\text{SR},\Pi}$ is defined as follows: (1) $\Psi^{1}_{(\text{SR},\Pi)}(S) = \Psi_{(\text{SR},\Pi)}(S) = S \cup \mathcal{R}_S(\Pi)$; (2) $\Psi^{i+1}_{(\text{SR},\Pi)}(S) = \Psi_{(\text{SR},\Pi)}\big(\Psi^{i}_{(\text{SR},\Pi)}(S)\big)$; and (3) $\Psi^{\infty}_{(\text{SR},\Pi)}(S) = \bigcup_{i=0}^{\infty} \Psi^{i}_{(\text{SR},\Pi)}(S)$.

As addressed in (Calautti et al., 2015a), using this definition, the class of SIZE-resticted programs strictly contains the class of GMT-bounded programs, and hence forms the decidable class of logic programs with function symbols containing all previous decidable classes.

Example 10. Assume Π to be the program consisting of the single rule:

$$r_1 : \mathsf{q}\big(f(X), h(Y), h(Z)\big) \leftarrow \mathsf{p}\big(X, g(Z,Z), g(Y,Y)\big),$$
$$r_2 : \mathsf{p}(X,Y,Z) \leftarrow \mathsf{q}(X,Y,Z).$$

Then since the cycle between the four arguments $\mathsf{p}[2]$, $\mathsf{p}[3]$, $\mathsf{q}[2]$ and $\mathsf{q}[3]$ are balanced, and where the cycle between the two arguments $\mathsf{p}[1]$ and $\mathsf{q}[1]$ is growing, then it follows from Definition 8 that Π cannot be GMT-bounded (unless if instead, the four arguments $\mathsf{p}[2]$, $\mathsf{p}[3]$, $\mathsf{q}[2]$ and $\mathsf{q}[3]$ have a failing cycle). On the other hand, since we get that $size\big(Hd(r_1)\big) = (1+x, 1+y, 1+z)$ and $size(B) = (x,\ 2+2z,\ 2+2y)$, where $B \in Pos(r_1) = \big\{\mathsf{p}\big(X, g(Z,Z), g(Y,Y)\big)\big\}$, then under the vector $\overline{v} = (1,1,1)$, we get that $\overline{v} \cdot size(B) \geq \overline{v} \cdot size\big(Hd(r_1)\big)$ holds for all the non-negative integer values of the variables mentioned in the vectors $size\big(Hd(r_1)\big)$ and $size(B)$. Therefore, it follows that Π is SIZE-restricted. □

Appendix B. Proofs

B.1 Proof of Theorem 1

Theorem 1. *Let Π be a program and $S \subseteq \mathcal{L}^{\infty}(\Pi)$. Then for every input database D and stable model M of $\Pi \cup D$, $M \ll \mathcal{U}^{\infty}(\Pi, S)$.*

Proof. Suppose that M is a stable model of $\Pi' = \text{GROUND}(\Pi \cup D)$ such that $M \not\ll \mathcal{U}^{\infty}(\Pi, S)$. Then since M is a stable model of Π', it follows that M is a minimal model of $(\Pi')^M$ $\big($where $(\Pi')^M$ denotes the reduct of Π' on $M\big)$. Let $M' = M \setminus \{a \in M \mid a \not\ll \mathcal{U}^{\infty}(\Pi, S)\}$, i.e., M' is the set obtained from M by deleting all the atoms that are not embeddable in $\mathcal{U}^{\infty}(\Pi, S)$. Then since $M \not\ll \mathcal{U}^{\infty}(\Pi, S)$ (which implies that there exists some $a \in M$ such that $a \not\ll \mathcal{U}^{\infty}(\Pi, S)$), it follows that $M' \subset M$ where $M' \ll \mathcal{U}^{\infty}(\Pi, S)$.

Lemma 1. *$M' \models (\Pi')^M$.*

Proof. On the contrary, we assume that $\exists r \in (\Pi')^M$ such that $Pos(r) \subseteq M'$ where $M' \cap Hd(r) = \emptyset$. Then since all rules in $(\Pi')^M$ only have positive bodies and $M' \ll \mathcal{U}^{\infty}(\Pi, S)$, it follows by backtracking that all the atoms in $Pos(r)$ are embeddable into $\mathcal{U}^{k}(\Pi, S)$, for some stage k, i.e., $Pos(r) \ll \mathcal{U}^{k}(\Pi, S)$, for some $k \geq 0$. Moreover, since $Neg(r) \cap S^{+}\restriction_{\text{CONST}(\Pi \cup D)} = \emptyset$ [11] (since $r \in (\Pi')^M$ and $S^{+}\restriction_{\text{CONST}(\Pi \cup D)} \subseteq M$ by Proposition 2) and $Pos(r) \cap S^{-}\restriction_{\text{CONST}(\Pi \cup D)} = \emptyset$ (also by Proposition 2 and since $Pos(r) \subseteq M' \subset M$), then from the definition of $\mathcal{U}^{k+1}(\Pi, S)$ (see Definition 3), it follows that $Hd(r) \ll \mathcal{U}^{k+1}(\Pi, S)$. On the other hand, from the definition of M', it implies $M' \cap Hd(r) \neq \emptyset$ due to $M \models (\Pi')^M$, $Pos(r) \subseteq M' \subset M$ and $Neg(r) \cap M = \emptyset$ (i.e., $Hd(r) \in M$). Obviously, this contradicts the assumption that $M' \cap Hd(r) = \emptyset$. □

Therefore, from Lemma 1, we get a contradiction that M is a minimal model of $(\Pi')^M$. □

B.2 Proof of Proposition 7

Proposition 7. *Given a program Π, $\text{AR}(\Pi) \subseteq \text{polyLA}(\Pi)$.*

Proof. We prove this result for $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)$ where $S = \emptyset$. It then follows from Proposition 4 that the result also holds for the case where $S = \mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$.

On the contrary, assume that there exists some $p[i] \in \text{AR}(\Pi)$ such that $p[i] \notin \text{polyLA}(\Pi)$. Then from the definition of $\text{polyLA}(\Pi)$ we get $\text{DEP}_{p[i]}\big(\mathcal{U}^{2 \cdot \mathbb{P}(\Pi)}(\Pi, \emptyset)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{3 \cdot \mathbb{P}(\Pi)}(\Pi, \emptyset)\big)$. First, the key thing to observe here is that there can only be utmost $P = |\Pi^{\text{DEF}}| \cdot \text{maxArt}(\Pi) \cdot \text{maxPos}(\Pi) < \mathbb{P}(\Pi)$ different positions that argument $p[i]$ can be propagated in Π^{DEF}. Now, the first iterations comprising of P-steps considers the possibility that $p[i]$

11. For convenience, we assume that $Neg(r)$ stands for the negative body of r prior to its reduced form in the reduct $(\Pi')^M$.

could skip a propagation onto itself. However, by the $2 \cdot P + 1$-step, we would have already considered all the possible propagations of $p[i]$. Therefore by the *pigeonhole principle*, where we view P here as the number of holes and the number of iteration steps as the pigeons, it follows that $p[i]$ would have been propagated onto itself if it still grows a complex term by that stage (because even growth due to the successive applications of rules alone would have been considered in the first P-steps). In fact, since even at stage $\mathcal{U}^{3 \cdot \mathbb{P}(\Pi)}(\Pi)$, where $3 \cdot P < 3 \cdot \mathbb{P}(\Pi)$, the argument $p[i]$ still contains a complex term that grows in depth, then $p[i]$ must be in a growing cycle, which contradicts the assumption that $p[i] \in \text{AR}(\Pi)$. □

B.3 Proof of Theorem 2

Theorem 2. *If Π is* POLY*-bounded, then for every input database D (D can be empty), program $\Pi \cup D$ is finitely ground.*

Proof. Before we get into the main proof of Theorem 2, we first introduce the following lemma whose result is going to be used in our main proof.

Lemma 2. *Given a program Π, some $S \subseteq \mathcal{L}^{\infty}(\Pi)$, some rule $r \in \Pi^{\text{DEF}}$ and assignment $\theta' : \text{VAR}(r) \longrightarrow \text{TERMS}\big(\mathcal{U}^{\infty}(\Pi, S)\big)$, if for some $k > \mathbb{P}(\Pi)$ we have that $Pos(r)\theta' \subseteq \mathcal{U}^{k-1}(\Pi, S)$ such that $Hd(r)\theta' \in \mathcal{U}^{k}(\Pi, S)$ and $Hd(r)\theta' = B\eta'$, for some $B \in \text{POS}(\Pi)$ and corresponding assignment η', then there also exists some assignment $\theta : \text{VAR}(r) \longrightarrow \text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}}) \cup \mathcal{U}^{i-1}(\Pi, S)\big)$ such that $Hd(r)\theta \in \mathcal{U}^{i}(\Pi, S)$, and where: (1) $1 \leq i \leq K$; (2) $r \in \text{SCC}(\Pi^{\text{DEF}})^{[i]}$; (3) $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$, $\text{SCC}(\Pi^{\text{DEF}})^{[1]}$, …, $\text{SCC}(\Pi^{\text{DEF}})^{[i]}$, …, $\text{SCC}(\Pi^{\text{DEF}})^{[K]}$, corresponds to the SCC stratification of $\text{SCC}(\Pi^{\text{DEF}})$ (see Section 2.7); and (4) $Hd(r)\theta = B\eta$, for some corresponding assignment η.*

Proof. Let $Hd(r)\theta' = A \in \mathcal{U}^{k}(\Pi, S) \big\backslash \mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)$ (for some $k > \mathbb{P}(\Pi)$), $B \in \text{POS}(\Pi)$ and $\eta' : \text{VAR}(B) \longrightarrow \text{TERMS}(A)$ such that $B\eta' = A$. Then since $A \in \mathcal{U}^{k}(\Pi, S) \big\backslash \mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)$ $\big($which implies that $A \notin \mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)\big)$, then given the size of the bound $\mathbb{P}(\Pi)$ $\big($see (35)$\big)$, then it follows that the rule $r \in \Pi^{\text{DEF}}$ is "recursive" in the sense that $recrPos(r) \neq \emptyset$, due to (22) of Definition 3. On the other hand, because of (21) of Definition 3, that recursive rule $r \in \Pi^{\text{DEF}}$ would have also been applied at some step $\mathcal{U}^{i}(\Pi, S)$, where $r \in \text{SCC}(\Pi^{\text{DEF}})^{[i]}$, and such that: $\text{SCC}(\Pi^{\text{DEF}})^{[0]}$, $\text{SCC}(\Pi^{\text{DEF}})^{[1]}$, …, $\text{SCC}(\Pi^{\text{DEF}})^{[i]}$, …, $\text{SCC}(\Pi^{\text{DEF}})^{[K]}$, corresponds to the SCC stratification of $\text{SCC}(\Pi^{\text{DEF}})$ (i.e., $Hd(r)\theta = A' \in \mathcal{U}^{i}(\Pi, S)$ for some assignment θ) and $K \leq \mathbb{P}(\Pi)$. Then, assuming that $Hd(r)\theta = A'$, since we get from (21) that the assignment $\theta : \text{VAR}(r) \longrightarrow \text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}}) \cup \mathcal{U}^{i-1}(\Pi, S)\big)$ considers all terms and subterms of all the atoms in $\text{ATOMS}(\Pi^{\text{DEF}})$, then it follows that through such the assignment θ, we can unify A' and B as well, i.e., there exists another assignment $\eta : \text{VAR}(B) \longrightarrow \text{TERMS}(A')$ such that $B\eta = A'$. This completes the proof of Lemma 2. □

Intuitively, Lemma 2 says that the atoms contained in $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)$ are enough to consider all the possible "types" and "subtypes" of terms that can be derived in the heads of rules in Π^{DEF} and up to whose terms depth are not greater than $\text{MAXDEP}(\Pi)$ (because the assignment θ, as mentioned in the proof above, considers all the assignment of $\text{VAR}(r)$ onto

$\text{TERMS}\big(\text{ATOMS}(\Pi^{\text{DEF}})\big)$ in the step $\mathcal{U}^i(\Pi, S)$). Now we are ready to proceed to the main proof of Theorem 2.

We prove this main result for $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)$ where $S = \emptyset$. It then follows from Proposition 4 that the result also holds for the case where $S = \mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$.

Let $p[i] \in \text{POLYLA}(\Pi)$. Then based on the definition of $\text{POLYLA}(\Pi)$ as described in (36), we have that the following condition holds:

$$\text{DEP}_{p[i]}\big(\mathcal{U}^{2\cdot\mathbb{P}(\Pi)}(\Pi, S)\big) = \text{DEP}_{p[i]}\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi, S)\big).$$

Then we have the following lemma.

Lemma 3. $\text{DEP}_{p[i]}\big(\mathcal{U}^{2\cdot\mathbb{P}(\Pi)}(\Pi, S)\big) = \text{DEP}_{p[i]}\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi, S)\big)$ *iff* $\text{DEP}_{p[i]}\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi, S)\big) = \text{DEP}_{p[i]}\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, S)\big)$ *holds for all* $k \geq 1$.

Proof. ("$\Longrightarrow$") On the contrary, we assume that there exists some $k \geq 1$ such that $\text{DEP}_{p[i]}\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, S)\big)$. Then now let us further prove the following lemma.

Lemma 4. *There exists a homomorphism*

$$h : \mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset) \longrightarrow 2^{\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)}$$

such that

$$\text{DEP}_{p[i]}\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big)$$

iff

$$\text{DEP}_{p[i]}\Big(h\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big)\Big) < \text{DEP}_{p[i]}\Big(h\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big)\Big),$$

where for $S \subseteq 2^{\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)}$, *we have* $h(S)$ *denotes the set of atoms*

$$\bigcup_{A\in S} h(A) \subseteq \mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset).$$

Before proceeding to the actual proof of Lemma 4, we first provide some explanation of the homomorphism $h : \mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset) \longrightarrow 2^{\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)}$. Intuitively, the homomorphism h is a mapping of the atoms in $\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)$ to the set of atoms in $2^{\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)}$ of the stage $3 \cdot \mathbb{P}(\Pi)$. Here, as indicated in the description of the lemma, the homomorphism h has the property that

$$\text{DEP}_{p[i]}\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big)$$

iff

$$\text{DEP}_{p[i]}\Big(h\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big)\Big) < \text{DEP}_{p[i]}\Big(h\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big)\Big),$$

i.e., loosely speaking, a term grows in depth from stage $(k+2) \cdot \mathbb{P}(\Pi)$ to $(k+3) \cdot \mathbb{P}(\Pi)$ iff it also grew in depth at stage $2 \cdot \mathbb{P}(\Pi)$ to $3 \cdot \mathbb{P}(\Pi)$ $\big($which contradicts the earlier assumption$\big)$. In particular, we note in the latter that the numbers $2 \cdot \mathbb{P}(\Pi)$ and $3 \cdot \mathbb{P}(\Pi)$ are also $(k+2) \cdot \mathbb{P}(\Pi)$ and $(k+3) \cdot \mathbb{P}(\Pi)$, respectively, when $k = 0$.

Proof. We construct the homomorphism by induction from $l = 0$ to $l = 2{\cdot}\mathbb{P}(\Pi)$ such that the property holds for each of the homomorphism $h_l : \mathcal{U}^{l+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \longrightarrow 2^{\left(\mathcal{U}^{l+\mathbb{P}(\Pi)}(\Pi,\emptyset)\right)}$. Clearly, we note here that when $l = 0$ and $l = 2{\cdot}\mathbb{P}(\Pi)$, then we will have that $l+(k+1)\cdot\mathbb{P}(\Pi) = (k+1)\cdot\mathbb{P}(\Pi)$ and $l+(k+1)\cdot\mathbb{P}(\Pi) = (k+3)\cdot\mathbb{P}(\Pi)$, respectively.

Basis: We map each atom $A \in \mathcal{U}^{(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)$ into $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi,\emptyset)$ via the homomorphism h_0 as follows:

$$
\begin{aligned}
&h_0(A) = \\
&\Big\{ Hd(r)\theta \mid r \in \Pi^{\text{DEF}} \text{ and } \theta : \textsc{var}(r) \longrightarrow \textsc{varConst}(\Pi), \text{ and} \\
&\qquad \exists j \in \{0,\ldots,(k+1)\cdot\mathbb{P}(\Pi)-1\} \text{ and corresponding} \\
&\qquad \text{assignment } \theta' : \textsc{var}(r) \longrightarrow \textsc{terms}\big(\mathcal{U}^j(\Pi,\emptyset)\big) \text{ s.t.} \\
&\qquad Pos(r)\theta' \subseteq \mathcal{U}^j(\Pi,\emptyset),\ Hd(r)\theta' \in \mathcal{U}^{j+1}(\Pi,\emptyset) \text{ and} \\
&\qquad Hd(r)\theta' = A \Big\}.
\end{aligned}
$$

Intuitively, each atom $A \in \mathcal{U}^{(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)$ is mapped into the set of all the possible atom types mentioned in any head of a rule $r \in \Pi^{\text{DEF}}$ that can be used to derive A in some stage $\mathcal{U}^j(\Pi,\emptyset)$ $\big($for $j \in \{0, \ldots, (k+1)\cdot\mathbb{P}(\Pi)-1\}\big)$. More precisely, we get from Lemma 2 that $h_0\big(A\big) \subseteq \mathcal{U}^{\mathbb{P}(\Pi)}(\Pi,\emptyset)$ because the stages $\mathcal{U}^j(\Pi,\emptyset)$ $\big($for $j \in \{1, \ldots, \mathbb{P}(\Pi)\}\big)$ would have derived all such atom types.

Inductive step: Assume that we had defined the homomorphism $h_j : \mathcal{U}^{j+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \longrightarrow 2^{\mathcal{U}^{j+\mathbb{P}(\Pi)}(\Pi,\emptyset)}$ for $j \in \{0,\ldots,l\}$.

Then for each atom $A \in \big(\mathcal{U}^{(l+1)+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \setminus \mathcal{U}^{l+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)$, we further define $h_{l+1}(A)$ to be the following set of atoms such that:

$$
\begin{aligned}
&h_{l+1}(A) = \\
&\Big\{ Hd(r)\theta \mid r \in \Pi^{\text{DEF}} \text{ and } \theta : \textsc{var}(r) \longrightarrow \textsc{term}\big(\mathcal{U}^{l+\mathbb{P}(\Pi)}(\Pi,\emptyset)\big) \text{ such that:} \\
&\qquad (1)\ \exists\theta' : \textsc{var}(r) \longrightarrow \textsc{terms}\big(\mathcal{U}^{l+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big) \\
&\qquad\quad \text{s.t. } Pos(r)\theta' \subseteq \mathcal{U}^{l+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset), \\
&\qquad\quad Hd(r)\theta' \in \mathcal{U}^{(l+1)+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \text{ and } Hd(r)\theta' = A; \\
&\qquad (2)\ recrPos(r)\theta \subseteq \bigcup_{B \in recrPos(r)} h_l(B) \text{ and} \\
&\qquad\quad \big(Pos(r) \setminus recrPos(r)\big)\theta \subseteq \mathcal{U}^{l+\mathbb{P}(\Pi)}(\Pi,\emptyset) \Big\}.
\end{aligned}
$$

Intuitively, $h_{l+1} : \mathcal{U}^{(l+1)+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \longrightarrow 2^{\mathcal{U}^{(l+1)+\mathbb{P}(\Pi)}(\Pi,\emptyset)}$ is an extension of the previous one $h_l : \mathcal{U}^{l+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \longrightarrow 2^{\mathcal{U}^{l+\mathbb{P}(\Pi)}(\Pi,\emptyset)}$ by setting $h_{l+1}(A) = Hd(r)\theta$, for each $A \in \big(\mathcal{U}^{(l+1)+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \setminus \mathcal{U}^{l+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)$, and such that the corresponding positive body atoms are from those already mapped from the previous stage, i.e., $recrPos(r)\theta \subseteq \bigcup_{B\in recrPos(r)} h_l(B)$ and $\big(Pos(r) \setminus recrPos(r)\big)\theta \subseteq$

$\mathcal{U}^{l+\mathbb{P}(\Pi)}(\Pi,\emptyset)$, where $recrPos(r)$ denotes the mutually recursive atoms of $Pos(r)$ with $Hd(r)$ (see Section 2.7 for definition of "$recrPos(r)$"). Then since we have from the ind. hyp. that $\bigcup_{B\in recrPos(r)} h_l(B) \subseteq \mathcal{U}^{l+\mathbb{P}(\Pi)}(\Pi,\emptyset)$, then it follows from the definition of $\mathcal{U}^{(l+1)+\mathbb{P}(\Pi)}(\Pi,\emptyset)$ $\big($see (22) in Definition 3$\big)$ that $h_{l+1}\big(\mathcal{U}^{(l+1)+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \setminus \mathcal{U}^{l+(k+1)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big) \subseteq \mathcal{U}^{(l+1)+\mathbb{P}(\Pi)}(\Pi,\emptyset)$. Therefore, it follows that $h = h_{2\cdot\mathbb{P}(\Pi)}$ is well defined. Moreover, it further follows from the definition of the homomorphism h that $\textsc{dep}_{p[i]}\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big) < \textsc{dep}_{p[i]}\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)$ iff

$$\textsc{dep}_{p[i]}\Big(h\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)\Big) < \textsc{dep}_{p[i]}\Big(h\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)\Big).$$

This completes the proof of Lemma 4. □

Then our contradiction finally follows from the homomorphism $h : \mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \longrightarrow 2^{\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)}$ of Lemma 4 and since the size of the bound $\mathbb{P}(\Pi)$ $\big($see (35) for definition of $\mathbb{P}(\Pi)\big)$ would have already considered the growth of $p[i]$ in $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi,\emptyset)$ from the non-recursive application of rules in Π^{DEF}, and where the other cycles propagating complex terms in $\mathcal{U}^{2\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)$ would also propagate onto $\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)$. This ends the proof of Lemma 3.

("$\Longleftarrow$") Similarly, to prove this direction, we also use the homomorphism

$$h : \mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \longrightarrow 2^{\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)}$$

as was defined (constructively) in the proof Lemma 4. Indeed, assume by contradiction that for some $k \geq 1$, we have that $\textsc{dep}_{p[i]}\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi,S)\big) < \textsc{dep}_{p[i]}\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,S)\big)$ but where $\textsc{dep}_{p[i]}\big(\mathcal{U}^{2\cdot\mathbb{P}(\Pi)}(\Pi,S)\big) = \textsc{dep}_{p[i]}\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi,S)\big)$. Then we get a contradiction by the definition of the homomorphism $h : \mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset) \longrightarrow 2^{\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)}$ as defined constructively in the proof of Lemma 4 since the homomorphism "h" has the property that:

$$\textsc{dep}_{p[i]}\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big) < \textsc{dep}_{p[i]}\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)$$

iff

$$\textsc{dep}_{p[i]}\Big(h\big(\mathcal{U}^{(k+2)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)\Big) < \textsc{dep}_{p[i]}\Big(h\big(\mathcal{U}^{(k+3)\cdot\mathbb{P}(\Pi)}(\Pi,\emptyset)\big)\Big).$$

□

This ends the proof of Theorem 2. □

B.4 Proof of Theorem 3

Theorem 3. *The* poly*-bounded class strictly contains* gmt*-bounded class.*

Proof. ("$\subsetneq$") This follows from program Π of Example 3 that is poly-bounded but not gmt-bounded.

("$\subseteq$") We prove this result for $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi,S)$ where $S=\emptyset$. It then follows from Proposition 4 that the result also holds for the case where $S = \mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$.

We show that $\Psi^{\infty}_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big) \subseteq \text{POLY}\text{LA}(\Pi)$ by induction on i for $\Psi^{i}_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$.

Basis: Assume on the contrary that for some $q[l] \in \Psi^0_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$ we have that $q[l] \notin \text{POLYLA}(\Pi)$. Then by the definition of $\Psi^0_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$, we have that $q[l] \in \text{AR}(\Pi)$ because $\Psi^0_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big) = \text{AR}(\Pi)$. Therefore, we have from Proposition 7 that this is a contradiction.

Step: Assuming that $\Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big) \subseteq \text{POLYLA}(\Pi)$, let us also assume on the contrary that for some $q[l] \in \Psi^{i+1}_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big) \setminus \Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$, we have that $q[l] \notin \text{POLYLA}(\Pi)$. Then by the definition of $\text{POLYLA}(\Pi)$, it follows that $\text{DEP}_{q[l]}\big(\mathcal{U}^{2\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big) < \text{DEP}_{q[l]}\big(\mathcal{U}^{3\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)\big)$. Then we have that $q[l]$ is still growing in depth in the stages between $2 \cdot \mathbb{P}(\Pi)$ and $3 \cdot \mathbb{P}(\Pi)$, which indicates that $q[l]$ is in some basic cycle π in the argumentation graph $\mathcal{G}_L(\Pi)$. Moreover, since we also have that $q[l] \in \Psi^{i+1}_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big) \setminus \Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$, then we have that for each basic cycle π' that $q[l]$ depends on, at least one of the following conditions holds:

1. π' is not active or is not *growing*;
2. π' contains an edge $\big(s[j], p[i], \langle f, r, l_1, l_2\rangle\big)$ and letting $p(t_1, \ldots, t_n)$ be the l_1-th atom in the head of r, for every variable X in t_i, there is an atom $b(u_1, \ldots, u_m)$ in $Pos(r)$ such that X appears in a term u_h and $b[h] \in \Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$;
3. there is a basic cycle π'' in $\mathcal{G}_L(\Pi)$ such that $\pi'' \approx \pi'$, π'' is not balanced, and π'' only passes through arguments in $\Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$.

Then since $q[l]$ is still growing in $\mathcal{U}^{2\cdot\mathbb{P}(\Pi)}(\Pi, \emptyset)$, it follows that π is both active and growing, which implies that Item 1 above cannot apply to π. On the other hand, we have that Item 2 cannot apply to π either because nothing guarded it to have grown even up to the $2 \cdot \mathbb{P}(\Pi)^{th}$-stage, because all arguments $b[h] \in \Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$, as mentioned in Item 2, must have all grown as well, and since $\Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big) \subseteq \text{POLYLA}(\Pi)$ (by the ind. hyp.) Therefore, we must have that only Item 3 can apply. But then, because π'' only passes through $\Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$, and where we have $\Psi^i_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big) \subseteq \text{POLYLA}(\Pi)$ (ind. hyp.) and such that by the definition of $\text{POLYLA}(\Pi)$, we have that all the arguments in it are not growing past the $2 \cdot \mathbb{P}(\Pi)^{th}$-stage, then this contradicts that $\pi'' \approx \pi$ as well.

□

B.5 Proof of Theorem 4

Theorem 4. *The* POLY*-bounded class strictly contains the* SIZE*-restricted class.*

Proof. ("$\subsetneq$") This follows from program Π of Examples 3 and 8 that is POLY-bounded but not SIZE-restricted.

("$\subseteq$") We prove this result for $\mathcal{U}^{\mathbb{P}(\Pi)}(\Pi, S)$ where $S = \emptyset$. It then follows from Proposition 4 that the result also holds for the case where $S = \mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$.

Let $\mathcal{A} = \Psi^{\infty}_{(\text{GMT},\Pi)}\big(\text{AR}(\Pi)\big)$. Then, since $\text{AR}(\Pi) \subseteq \mathcal{A}$, it will be sufficient to show that $\Psi^{\infty}_{(\text{SR},\Pi)}(\mathcal{A}) \subseteq \text{POLYLA}(\Pi)$, which we do by induction on k for $\Psi^{k}_{(\text{SR},\Pi)}(\mathcal{A})$.

Basis: Since $\Psi^{1}_{(\text{SR},\Pi)}(\mathcal{A}) = \mathcal{A} \cup \mathcal{R}_{\mathcal{A}}(\Pi)$, we show that $\mathcal{A} \cup \mathcal{R}_{\mathcal{A}}(\Pi) \subseteq \text{POLYLA}(\Pi)$. For the sake of contradiction, assume that there exists some $p[i] \in \mathcal{A} \cup \mathcal{R}_{\mathcal{A}}(\Pi)$ $\big($where $i \in \{1, \ldots, \text{ARITY}(p)\}$ $\big)$ such that $p[i] \notin \text{POLYLA}(\Pi)$. Then since we had by Theorem 3 that $\mathcal{A} \subseteq \text{POLYLA}(\Pi)$, then $p[i] \notin \mathcal{A}$ and $p[i] \in \mathcal{R}_{\mathcal{A}}(\Pi)$. Now, since $p[i] \notin \text{POLYLA}(\Pi)$, then this implies that there exists some $p[i] \in \text{ARG}(\Pi)$ such that $\big(\mathcal{U}^{2 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{3 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big)$. Then it follows by Lemma 3 as used in the proof of Theorem 2 that $\text{DEP}_{p[i]}\big(\mathcal{U}^{(k+2) \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{(k+3) \cdot \mathbb{P}(\Pi)}(\Pi, S)\big)$ holds for all $k \geq 1$. Then clearly, this implies that for: $k = 1$; $k = 2$; $k = 3$; ..., we get that:

$$\text{DEP}_{p[i]}\big(\mathcal{U}^{3 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{4 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{5 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \ldots.$$

Therefore, since the depth of the terms mentioned in the argument $p[i]$ continues to grow indefinitely, then it follows that this contradicts that $p[i] \in \mathcal{A} \cup \mathcal{R}_{\mathcal{A}}(\Pi)$.

Step: Assume we have $\Psi^{j}_{(\text{SR},\Pi)}(\mathcal{A}) \subseteq \text{POLYLA}(\Pi)$ for $j \in \{1, \ldots, k\}$. Now assume on the contrary that for some $p[i] \in \big(\Psi^{k+1}_{(\text{SR},\Pi)}(\mathcal{A}) \setminus \Psi^{k}_{(\text{SR},\Pi)}(\mathcal{A})\big)$, we have that $p[i] \notin \text{POLYLA}(\Pi)$. Then this again implies that there exists some $p[i] \in \text{ARG}(\Pi)$ such that $\big(\mathcal{U}^{2 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{3 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big)$. Then it follows again by Lemma 3 as was used in the proof of Theorem 2 that $\text{DEP}_{p[i]}\big(\mathcal{U}^{(k+2) \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{(k+3) \cdot \mathbb{P}(\Pi)}(\Pi, S)\big)$ holds for all $k \geq 1$. Then clearly, this implies that for: $k = 1$; $k = 2$; $k = 3$; ..., we get that:

$$\text{DEP}_{p[i]}\big(\mathcal{U}^{3 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{4 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \text{DEP}_{p[i]}\big(\mathcal{U}^{5 \cdot \mathbb{P}(\Pi)}(\Pi, S)\big) < \ldots.$$

Then again, as used in the base case above, since the depth of the terms mentioned in the argument $p[i]$ continues to grow indefinitely, then it follows that this contradicts that $p[i] \in \big(\Psi^{k+1}_{(\text{SR},\Pi)}(\mathcal{A}) \setminus \Psi^{k}_{(\text{SR},\Pi)}(\mathcal{A})\big)$

□

B.6 Proof of Theorem 5

Theorem 5. *Given a program* Π *and some number* $k \geq 0$*, if* Π *is* k-EXP-bounded*, then for every input database* D *(*D *can be empty), program* $\Pi \cup D$ *is finitely ground.*

Proof. The proof follows in the same way we did for Theorem 2 and Lemma 3 only this time, we set the bound to $\mathsf{exp}_{\Pi}(k)$, as defined in the beginning of Section 5, instead of the (polynomial) bound $\mathbb{P}(\Pi)$ as we did for the POLY-bounded class. □

B.7 Proof of Theorem 6

Theorem 6. *Deciding whether a program* Π *is* poly*-bounded is EXPTIME-complete. The hardness holds even if* Π*'s maximum function arity is 2.*

Proof. (*Membership*) Based on Proposition 9, we only need to consider condition (9) in Definition 4, and compute at most $3 \cdot \mathbb{P}(\Pi)$ iterations of the upper bound, i.e., $\mathcal{U}^{3 \cdot \mathbb{P}(\Pi)}(\Pi, S)$, to check if $\text{poly}\text{LA}(\Pi) = \text{arg}(\Pi)$, where $S = \mathcal{L}^{\mathbb{P}(\Pi)}(\Pi)$.

Then it is clear that the $\mathbb{P}(\Pi)$-steps of iterations for computing the lower bound and upper bound as described in Definitions 2 and 3, respectively, can be done in time

$$O(\overbrace{\mathbb{P}(\Pi)^{\mathbb{P}(\Pi)} \cdot \ldots \cdot \mathbb{P}(\Pi)^{\mathbb{P}(\Pi)}}^{\mathbb{P}(\Pi)\text{-times}}) = O(\mathbb{P}(\Pi)^{\mathbb{P}(\Pi)}).$$

Therefore, since $O(\mathbb{P}(\Pi)^{\mathbb{P}(\Pi)}) = O(2^k) \Longrightarrow O\big(\log(\mathbb{P}(\Pi)^{\mathbb{P}(\Pi)})\big) = O\big(\log(2^k)\big) \Longrightarrow O\big((\mathbb{P}(\Pi)) \cdot \log(\mathbb{P}(\Pi))\big) = O(k) \Longrightarrow O(k) \leq O\big((\mathbb{P}(\Pi))^2\big)$. Then it follows that deciding if Π is poly-bounded can be done in time $O(2^{p(\mathbb{P}(\Pi))})$ for some polynomial $p(n)$.

(*Hardness*) Let L be an arbitrary decision problem in EXPTIME. Then from the definition of complexity class EXPTIME (Papadimitriou, 1994), there exists some *deterministic Turing machine* M such that for any string $\mathbf{s}$, $\mathbf{s} \in L$ iff M accepts $\mathbf{s}$ in at most $2^{p(|\mathbf{s}|)}$ steps for some polynomial $p(n)$. Consider a Turing machine M to be the tuple $\langle Q, \Gamma, \square, \Sigma, \delta, q_0, F\rangle$, where (1) $Q \neq \emptyset$ is a finite set of states; (2) $\Gamma \neq \emptyset$ is a finite set of alphabet symbols; (3) $\square \in \Gamma$ is the "blank" symbol; (4) $\triangleright \in \Gamma$ is the "left-end-marker" symbol; (5) $\Sigma \subseteq \Gamma \setminus \{\square, \triangleright\}$ is the set of input symbols; (6) $\delta : (Q \setminus F) \times \big(\Gamma \setminus \{\triangleright\}\big) \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function; (7) $q_0 \in Q$ is the initial state; and lastly, (8) $F = (F_{accept} \cup F_{reject}) \subseteq Q$ is the set of fina states such that $F_{accept} \cap F_{reject} = \emptyset$ and F_{accept} (F_{reject}) is the accepting (rejecting) states.

Now given a string $\mathbf{s} = a_0 \ldots a_{|\mathbf{s}|-1}$ such that $a_i \in \Sigma$ for $0 \leq i < |\mathbf{s}|$, we construct a program $\Pi_{M(\mathbf{s})} = \Pi^{\text{ORD}}_{M(\mathbf{s})} \cup \Pi^{\text{STR}}_{M(\mathbf{s})} \cup \Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})} \cup \Pi^{\text{ACCEPT}}_{M(\mathbf{s})} \cup \Pi^{\text{UNBOUND}}_{M(\mathbf{s})}$.

Program $\Pi^{\text{ORD}}_{M(\mathbf{s})}$ is to generate the linear ordering on the $p(|\mathbf{s}|)$-length tuples that will encode the computation time/steps as well as the individual cell-positions in the tape. In particular, $\Pi^{\text{ORD}}_{M(\mathbf{s})}$ is specified as follows:

$$\Pi^{\text{ORD}}_{M(\mathbf{s})} = \big\{\, num(n_i) \leftarrow \top \mid 0 \leq i \leq |\Gamma| - 1 \,\big\} \cup \tag{70}$$

$$\big\{\, num^{\text{MIN}}_k\big(\overbrace{n_0, \ldots, n_0}^{k\text{-times}}\big) \leftarrow \top \mid 0 \leq k \leq p(|\mathbf{s}|) - 1 \,\big\} \cup \tag{71}$$

$$\big\{\, num^{\text{MAX}}_k\big(\overbrace{n_{|\Gamma|-1}, \ldots, n_{|\Gamma|-1}}^{k\text{-times}}\big) \leftarrow \top \mid 0 \leq k \leq p(|\mathbf{s}|) - 1 \,\big\} \cup \tag{72}$$

$$\big\{\, n_i \prec n_{i+1} \leftarrow \top \mid 0 \leq i < |\Gamma| - 1 \,\big\} \cup \tag{73}$$

$$\big\{\, n_i < n_j \leftarrow \top \mid 0 \leq i < j \leq |\Gamma| - 1 \,\big\} \cup \tag{74}$$

$$\Big\{\, \mathbf{X}U \prec \mathbf{X}V \leftarrow U \prec V, num(X_1), \ldots, num(X_{p(|\mathbf{s}|)-1}) \mid \mathbf{X} = X_1 \ldots X_{p(|\mathbf{s}|)-1} \,\Big\} \cup \tag{75}$$

$$\Big\{\, \mathbf{X}U\mathbf{Y} \prec \mathbf{X}V\mathbf{Z} \leftarrow U \prec V, num(X_1), \ldots, num(X_l), num^{\text{MAX}}_{|\mathbf{Y}|}(\mathbf{Y}),\, num^{\text{MIN}}_{|\mathbf{Z}|}(\mathbf{Z}) \mid \mathbf{X} = X_1 \ldots X_l,\, |\mathbf{Y}| = |\mathbf{Z}| \text{ and } |\mathbf{X}| + |\mathbf{Z}| + 1 = p(|\mathbf{s}|) \,\Big\} \cup \tag{76}$$

$$\Big\{\, \mathbf{X}U\mathbf{Y} < \mathbf{X}V\mathbf{Z} \leftarrow U < V, num(X_1), \ldots, num(X_l), num(Y_1), \ldots, num(Y_m), num(Z_1), \ldots, num(Z_m) \mid \mathbf{X} = X_1 \ldots X_l,\, \mathbf{Y} = Y_1 \ldots Y_m,\, \mathbf{Z} = Z_1 \ldots Z_m \text{ and } l + m + 1 = p(|\mathbf{s}|) \,\Big\} \cup \tag{77}$$

$$\Big\{\, \mathbf{X}U\mathbf{Y} \leq \mathbf{X}V\mathbf{Z} \leftarrow U < V, num(X_1), \ldots, num(X_l), num(Y_1), \ldots, num(Y_m), num(Z_1), \ldots, num(Z_m) \mid \mathbf{X} = X_1 \ldots X_l,\, \mathbf{Y} = Y_1 \ldots Y_m,\, \mathbf{Z} = Z_1 \ldots Z_m \text{ and } l + m + 1 = p(|\mathbf{s}|) \,\Big\} \cup \tag{78}$$

$$\Big\{\, \mathbf{X} \leq \mathbf{X} \leftarrow num(X_1), \ldots, num(X_l) \mid \mathbf{X} = X_1 \ldots X_l \Big\}. \tag{79}$$

- The symbols n_i, for $i \in \{0, \ldots, |\Gamma| - 1\}$, are constant symbols and is used to denote the number i. So $n_1, \cdots, n_{|\Gamma|-1}$ denotes numbers 1, …, $|\Gamma| - 1$, respectively.

- The five sets (70)-(74) encode our numbering scheme so that the range from

  $$\underbrace{n_0 \ldots n_0}_{p(|\mathbf{s}|)\text{-times}} \text{ to } \underbrace{n_{|\Gamma|-1} \ldots n_{|\Gamma|-1}}_{p(|\mathbf{s}|)\text{-times}}$$

  spans at least $2^{p(|\mathbf{s}|)}$-steps. This is important because: $|\Gamma|^K = 2^{p(|\mathbf{s}|)} \Longrightarrow \log(|\Gamma|^K) = \log(2^{p(|\mathbf{s}|)}) \Longrightarrow K \cdot \log(|\Gamma|) = p(|\mathbf{s}|) \Longrightarrow K = \frac{p(|\mathbf{s}|)}{\log(|\Gamma|)} < p(|\mathbf{s}|)$. [12]

- For some $0 \leq i \leq |\Gamma|^{p(|\mathbf{s}|)}$, the (overlined-and-bolded) $\bar{\mathbf{i}}$ denotes the $p(|\mathbf{s}|)$-length tuple enumeration of the number i. For example,

  $$\begin{aligned}
  \overline{\mathbf{0}} &= \underbrace{n_0 \ldots n_0}_{(p(|\mathbf{s}|)-1)\text{-times}} n_0, \\
  \overline{\mathbf{1}} &= \underbrace{n_0 \ldots n_0}_{(p(|\mathbf{s}|)-1)\text{-times}} n_1, \\
  \overline{\mathbf{2}} &= \underbrace{n_0 \ldots n_0}_{(p(|\mathbf{s}|)-1)\text{-times}} n_2, \\
  &\vdots, \\
  \overline{|\mathbf{\Gamma}|^{\mathbf{p(|s|)}}} &= \underbrace{n_{|\Gamma|-1} \ldots n_{|\Gamma|-1}}_{p(|\mathbf{s}|)\text{-times}}.
  \end{aligned}$$

12. Without loss of generality, we assume $|\Gamma| \geq 2$ so that $\log(|\Gamma|) > 1$.

- "$\prec$," "$<$" and "$\leq$" as mentioned in rules (70)-(79) of $\Pi^{\text{ORD}}_{M(\mathbf{s})}$, denote the successor, less-than and less-than-or-equal relations, and their corresponding extensions to $p(|\mathbf{s}|)$-arity tuples using lexicographic ordering, respectively.

Program $\Pi^{\text{STR}}_{M(\mathbf{s})}$ generates all possible strings of lengths from 1 to $|\Gamma|^{p(|\mathbf{s}|)}$ under the alphabets of Γ:

$$
\begin{aligned}
\Pi^{\text{STR}}_{M(\mathbf{s})} = {} & \{\, s^0_0(\triangleright, \overline{\mathbf{0}}, \overline{\mathbf{0}}) \leftarrow \top \,\} \cup && (80)\\
& \{\, s^0_0\big(a_{i-1}, \bar{\mathbf{i}}, \bar{\mathbf{i}}\big) \leftarrow \top \mid 1 \leq i \leq |\mathbf{s}| \,\} \cup && (81)\\
& \{\, s^0_0\big(\square, \mathbf{T}, \mathbf{T}\big) \leftarrow \overline{\mathbf{n-1}} < \mathbf{T} \,\} \cup && (82)\\
& \{\, s^k_0(Z, \mathbf{T}_1, \mathbf{T}_4) \leftarrow s^i_0(X, \mathbf{T}_1, \mathbf{T}_2), s^j_0(Y, \mathbf{T}_3, \mathbf{T}_4), \\
& \qquad \mathbf{T}_1 \leq \mathbf{T}_2, \mathbf{T}_2 \prec \mathbf{T}_3, \mathbf{T}_3 \leq \mathbf{T}_4 \\
& \quad \mid Z \in \{\, (X \circ Y),\, (Y \circ X) \,\} \text{ and } 0 \leq i, j < k \leq p(|\mathbf{s}|) \,\} \cup && (83)\\
& \{\, s^0(\triangleright, \mathbf{T}, \mathbf{T}) \leftarrow num(T_1), \ldots, num(T_{p(|\mathbf{s}|)}), \\
& \quad s^0\big(a, \mathbf{T}, \mathbf{T}\big) \leftarrow num(T_1), \ldots, num(T_{p(|\mathbf{s}|)}) \\
& \quad \mid a \in \Gamma \setminus \{\triangleright\} \text{ and } \mathbf{T} = T_1 \ldots T_{p(|\mathbf{s}|)} \,\} \cup && (84)\\
& \{\, s^k(Z, \mathbf{T}_1, \mathbf{T}_4) \leftarrow s^i(X, \mathbf{T}_1, \mathbf{T}_2), s^j(Y, \mathbf{T}_3, \mathbf{T}_4), \\
& \qquad \mathbf{T}_1 \leq \mathbf{T}_2, \mathbf{T}_2 \prec \mathbf{T}_3, \mathbf{T}_3 \leq \mathbf{T}_4 \\
& \quad \mid Z \in \{\, (X \circ Y),\, (Y \circ X) \,\} \text{ and } 0 \leq i, j < k \leq p(|\mathbf{s}|) \,\} \cup && (85)\\
& \{\, s^i_0\big(X \circ (Y \circ Z), \mathbf{T}_1, \mathbf{T}_4\big) \leftarrow s^i_0\big((X \circ Y) \circ Z, \mathbf{T}_1, \mathbf{T}_4\big), \\
& \quad s^i_0\big((X \circ Y) \circ Z, \mathbf{T}_1, \mathbf{T}_4\big) \leftarrow s^i_0\big(X \circ (Y \circ Z), \mathbf{T}_1, \mathbf{T}_4\big), \\
& \quad s^i\big(X \circ (Y \circ Z), \mathbf{T}_1, \mathbf{T}_4\big) \leftarrow s^i\big((X \circ Y) \circ Z, \mathbf{T}_1, \mathbf{T}_4\big), \\
& \quad s^i\big((X \circ Y) \circ Z, \mathbf{T}_1, \mathbf{T}_4\big) \leftarrow s^i\big(X \circ (Y \circ Z), \mathbf{T}_1, \mathbf{T}_4\big) \\
& \quad \mid 0 \leq i \leq p(|\mathbf{s}|)\} \cup && (86)\\
& \{\, s_0(X, \mathbf{T}_1, \mathbf{T}_2) \leftarrow s^i_0(X, \mathbf{T}_1, \mathbf{T}_2), \\
& \quad s(X, \mathbf{T}_1, \mathbf{T}_2) \leftarrow s^i(X, \mathbf{T}_1, \mathbf{T}_2) \mid 0 \leq i \leq p(|\mathbf{s}|) \,\} \cup && (87)\\
& \{\, X \cong X \leftarrow s\big(X, \mathbf{T}_1, \mathbf{T}_2\big), \\
& \quad X \circ Y \cong X' \circ Y' \leftarrow s\big(X \circ Y, \mathbf{T}_1, \mathbf{T}_2\big),\, s\big(X' \circ Y', \mathbf{T}_1, \mathbf{T}_2\big), \\
& \qquad X \cong X',\, Y \cong Y', \\
& \quad (X \circ Y) \circ Z \cong X' \circ (Y' \circ Z') \leftarrow s\big((X \circ Y) \circ Z, \mathbf{T}_1, \mathbf{T}_2\big), \\
& \qquad s\big(X' \circ (Y' \circ Z'), \mathbf{T}_1, \mathbf{T}_2\big), \\
& \qquad X \cong X',\, Y \cong Y',\, Z \cong Z' \,\}, && (88)
\end{aligned}
$$

where:

- For each alphabet $a \in \Gamma \setminus \{\triangleright\}$, we have that the constant symbol "a" denotes the alphabet "a" and $\square$ is the blank symbol;

Without loss of generality, we assume $|\Gamma| > 2$, therefore, it is sufficient to use strings of length from 1 to $|\Gamma|^{p(|\mathbf{s}|)}$ to encode all possible $2^{p(|\mathbf{s}|)}$ $M(\mathbf{s})$ tape configurations.

In program $\Pi^{\text{STR}}_{M(\mathbf{s})}$, we define the function "$\circ$" which has arguments as two strings $\mathbf{s}_1$ and $\mathbf{s}_2$ so that $\mathbf{s}_1 \circ \mathbf{s}_2$ denotes the concatenation of $\mathbf{s}_1$ and $\mathbf{s}_2$. Then due to the transitivity rules (83) and (85), it follows that it would only take $O\big(p(|\mathbf{s}|)\big)$-steps to generate all such strings of lengths from 1 to $|\Gamma|^{p(|\mathbf{s}|)}$. Here, predicate s_0 is used to represent the input string on the tape, and predicates s^i_0 $\big($for $i \in \{0, \ldots, p(|\mathbf{s}|)\}$ and such that each s^i_0 are different predicates$\big)$ are used to generate the initial string; while predicate s represents an arbitrary string on the tape, which is generated from predicates s^i $\big($for $i \in \{0, \ldots, p(|\mathbf{s}|)\}$ and such that each s^i are different predicates$\big)$. We further note here that the last set of rules in (86) is simply for the closure of the associative property of the strings. Lastly, the last three rules in (88) is an inductive definition for deriving the associativity relations "$\cong$" between the strings.

Before proceeding to the definition of the other programs, we first introduce the following Lemma 5 which shows that the program $\Pi^{\text{STR}}_{M(\mathbf{s})}$ indeed generates all the strings of length from 1 to $|\Gamma|^{p(|\mathbf{s}|)}$ corresponding to the tape configuration of the machine M. First we denote that from here on, given some complex term $\mathbf{x}$ such that $\mathbf{x}$ is mentioned in some atom of the form "$s(\mathbf{x}, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2)$, " we denote by $\mathsf{str}(\mathbf{x})$ as the string corresponding to the (complex) term $\mathbf{x}$ in the natural way. For instance, if (say) $\mathbf{x} = \big((a \circ b) \circ (b \circ c)\big) \circ d$, then $\mathsf{str}(\mathbf{x})$ denotes string "*abbcd*. "

Lemma 5. *Let* $\mathbf{x}$ *be a string of symbols from* Γ *such that* $1 \leq |\mathbf{x}| \leq 2^{p(|\mathbf{s}|)}$*, and* n_1 *and* n_2 *two numbers describing tape sections s.t.* $0 \leq n_1 < n_2 \leq 2^{p(|\mathbf{s}|)}$ *and* $n_2 - n_1 = |\mathbf{x}|$*. Then there exists some* $s(\mathbf{y}, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ *such that* $\mathsf{str}(\mathbf{y}) = \mathbf{x}$*.*

Proof. We prove by induction on i for $i = 0$ to $i = p(|\mathbf{s}|)$ that the set of strings

$$S^i_{(n_1,n_2)} = \big\{\, \mathsf{str}(\mathbf{y}) \mid s^i(\mathbf{y}, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+ \text{ and } 1 \leq k \leq 2^i \big\} \tag{89}$$

contains all strings $\mathbf{x}$ s.t. $1 \leq |\mathbf{x}| \leq 2^i$ and contained in tape positions $\overline{\mathbf{n}}_1$ to $\overline{\mathbf{n}}_2$, where $n_2 - n_1 = |\mathbf{x}|$. Indeed, it is not too difficult to see that this holds for the base case when $i = 0$ via the rule (85). Thus, assume now that $S^j_{(n_1,n_2)}$, for $1 \leq j \leq i$, contains all strings of lengths 1 to 2^i from any arbitrary tape positions $\overline{\mathbf{n}}_1$ to $\overline{\mathbf{n}}_2$. Now let $\mathbf{x}$ be any arbitrary string of symbols from Γ s.t. $2^i < |\mathbf{x}| \leq 2^{i+1}$ and n_2, $n_1 \in \{1, \ldots, p(|\mathbf{s}|)\}$ be two numbers such that $n_2 - n_1 = |\mathbf{x}|$. Then it follows that $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2$ such that $1 \leq |\mathbf{x}_1| \leq 2^i$ and $1 \leq |\mathbf{x}_2| \leq 2^i$, and there exists another number $n_1 < m < n_2$ such that $m - n_1 = |\mathbf{x}_1|$ and $n_2 - (m + 1) = |\mathbf{x}_2|$. Then by the ind. hyp., we have that $\mathbf{x}_1$ and $\mathbf{x}_2$ are both contained in the sets $S^i_{(n_1,m)}$ and $S^i_{(m+1,n_2)}$, respectively. Therefore, via the rules (87) above, it follows that $\mathbf{x}$ will be in the set of strings $S^{i+1}_{(n_1,n_2)}$ as well. □

Programs $\Pi^{\text{EDGES}}_{M(\mathbf{s})}$ and $\Pi^{\text{TRANS}}_{M(\mathbf{s})}$ described below then encodes the machine $M(\mathbf{s})$'s configuration changes based on the corresponding state transitions in $M(\mathbf{s})$, for that we view that the input string $\mathbf{s}$ is accepted by machine $M(\mathbf{s})$ as the problem of reachability from the

initial configuration of $M(\mathbf{s})$ to $M(\mathbf{s})$'s final (accepting) configuration.

$$\begin{aligned}
&\Pi^{\text{EDGES}}_{M(\mathbf{s})} = \\
&\big\{\, cf\big(\mathbf{X}_t, q, X \circ a, \overline{\mathbf{0}}, \mathbf{X}_{tp} \,\|\, c \circ Y, \mathbf{Y}_{tp}, \overline{\mathbf{N}}\big) \vdash cf\big(\mathbf{Y}_t, q', X' \circ c, \overline{\mathbf{0}}, \mathbf{Y}_{tp} \,\|\, d \circ Y', \mathbf{Z}_{tp}, \overline{\mathbf{N}}\big) \\
&\qquad \leftarrow \mathbf{X}_t \prec \mathbf{Y}_t, \mathbf{X}_{tp} \prec \mathbf{Y}_{tp}, \mathbf{Y}_{tp} \prec \mathbf{Z}_{tp}, \\
&\qquad\quad s\big(X \circ a, \overline{\mathbf{0}}, \mathbf{X}_{tp}\big),\ s\big(c \circ Y, \mathbf{Y}_{tp}, \overline{\mathbf{N}}\big),\ s\big((X \circ b) \circ c, \overline{\mathbf{0}}, \mathbf{Y}_{tp}\big),\ s(Y, \mathbf{Z}_{tp}, \overline{\mathbf{N}}), \\
&\qquad\quad X' \cong X \circ b,\ Y \cong d \circ Y' \\
&\quad \mid\ \delta(q, a) = (q', b, R) \text{ and } c, d \in \Gamma, \text{ and } N = |\Gamma|^{p(|\mathbf{s}|)} \,\big\} \cup
\end{aligned} \tag{90}$$

$$\begin{aligned}
&\big\{\, cf\big(\mathbf{X}_t, q, X \circ c, \overline{\mathbf{0}}, \mathbf{X}_{tp} \,\|\, a \circ Y, \mathbf{Y}_{tp}, \overline{\mathbf{N}}\big) \vdash cf\big(\mathbf{Y}_t, q', X' \circ d, \overline{\mathbf{0}}, \mathbf{Z}_{tp} \,\|\, c \circ Y', \mathbf{X}_{tp}, \overline{\mathbf{N}}\big) \\
&\qquad \leftarrow \mathbf{X}_t \prec \mathbf{Y}_t, \mathbf{Z}_{tp} \prec \mathbf{X}_{tp}, \mathbf{X}_{tp} \prec \mathbf{Y}_{tp}, \\
&\qquad\quad s\big((X \circ c) \circ a, \overline{\mathbf{0}}, \mathbf{X}_{tp}\big),\ s(Y, \mathbf{Y}_{tp}, \overline{\mathbf{N}}),\ s\big(X \circ c, \overline{\mathbf{0}}, \mathbf{Z}_{tp}\big),\ s\big(b \circ Y, \mathbf{X}_{tp}, \overline{\mathbf{N}}\big), \\
&\qquad\quad X \cong X' \circ d,\ Y' \cong b \circ Y \\
&\quad \mid\ \delta(q, a) = (q', b, L) \text{ and } c, d \in \Gamma, \text{ and } N = |\Gamma|^{p(|\mathbf{s}|)} \,\big\};
\end{aligned} \tag{91}$$

$$\begin{aligned}
\Pi^{\text{TRANS}}_{M(\mathbf{s})} = \big\{\, & cf(\mathbf{X}) \Vdash^0 cf(\mathbf{Y}) \leftarrow cf(\mathbf{X}) \vdash cf(\mathbf{Y}), \\
& cf(\mathbf{X}) \Vdash^k cf(\mathbf{Z}) \leftarrow cf(\mathbf{X}) \Vdash^i cf(\mathbf{Y}),\ cf(\mathbf{Y}) \Vdash^j cf(\mathbf{Z}) \\
& \quad \mid\ 0 \le i, j < k \le p(|\mathbf{s}|) \big\}.
\end{aligned} \tag{92}$$

Here notation "$cf\big(\mathbf{X}_t, q, X \circ c, \overline{\mathbf{0}}, \mathbf{X}_{tp} \,\|\, a \circ Y, \mathbf{Y}_{tp}, \overline{\mathbf{N}}\big)$" mentioned in (90), and denoted as "$cf(\overline{\mathbf{X}})$" in (92), represents the machine's configuration. More precisely, assuming $\delta(q, a) = (q', b, R)$, the expression

$$\text{“ } cf\big(\mathbf{X}_t, q, X \circ c, \overline{\mathbf{0}}, \mathbf{X}_{tp} \,\|\, a \circ Y, \mathbf{Y}_{tp}, \overline{\mathbf{N}}\big) \vdash cf\big(\mathbf{Y}_t, q', X' \circ c, \overline{\mathbf{0}}, \mathbf{Y}_{tp} \,\|\, d \circ Y', \mathbf{Z}_{tp}, \overline{\mathbf{N}}\big), \text{ ”} \tag{93}$$

where $X' \cong X \circ b$ and $Y \cong d \circ Y'$ as mentioned in the head of the rules in (90), are relations[13] encoding the machine M's changes in configuration from time "$\mathbf{X}_t$" to (its successor) time "$\mathbf{Y}_t$" such that from the current (i.e., time "$\mathbf{X}_t$") configuration:

$$\begin{aligned}
&\text{“}_{X \circ a}\text{” is in tapes cells } \overline{\mathbf{0}}\text{–}\mathbf{X}_{tp} \text{ and cursor scanning “}_{a}\text{”} \\
&cf\big(\underbrace{\mathbf{X}_t}_{\text{time}}, \underbrace{q}_{\text{state}}, \overbrace{X \circ a, \overline{\mathbf{0}}, \mathbf{X}_{tp}}\ \|\ \underbrace{c \circ Y, \mathbf{Y}_{tp}, \overline{\mathbf{N}}}_{\text{“}_{c \circ Y}\text{” is in tapes cells } \mathbf{Y}_{tp}\text{–}\overline{\mathbf{N}}}\big),
\end{aligned} \tag{94}$$

the next configuration is

$$cf\big(\mathbf{Y}_t, q', X' \circ c, \overline{\mathbf{0}}, \mathbf{Y}_{tp} \,\|\, d \circ Y', \mathbf{Z}_{tp}, \overline{\mathbf{N}}\big),$$

such that $X' \cong X \circ b$ and $Y \cong d \circ Y'$ holds. In particular, in regards to (94), we note here that the "$\|$" simply puts the intuition that the head/cursor is separating the machine M's

13. For better presentation, we expressed (93) in this way so that it is more intuitive with the traditional configuration changes of a Turing machine, although we note that it is assumed to be expressed under one predicate symbol of appropriate arguments.

tape into two parts: "$X \circ a$" for $\overline{\mathbf{0}}$ to $\mathbf{X}_{tp}$ (and where the head is scanning "a") and "$c \circ Y$" for $\mathbf{Y}_{tp}$ to $\overline{\mathbf{N}}$, and such that $\mathbf{X}_{tp} \prec \mathbf{Y}_{tp}$ (i.e., $\mathbf{Y}_{tp}$ is the successor cell position of $\mathbf{X}_{tp}$) and $N = |\Gamma|^{p(|\mathbf{s}|)}$ (i.e., up to the last tape cell position).

The expression "$cf(\mathbf{X}) \Vdash^i cf(\mathbf{Y})$" $\big($for $i \in \{0, \ldots, p(|\mathbf{s}|)\}\big)$ in (92) encodes the transitive extension of "$\vdash$", for which it is read: configuration "$cf(\mathbf{Y})$" is reached from configuration "$cf(\mathbf{X})$". In particular, we note that the tuple of terms $\overline{\mathbf{X}}$ and $\mathbf{Y}$ does not contain any complex terms and are all variables. Here, for each pair $i, j \in \{0, \ldots, p(|\mathbf{s}|)\}$ such that $i \neq j$, we have that two atoms $cf(\mathbf{X}) \Vdash^i cf(\mathbf{Y})$ and $cf(\mathbf{X}) \Vdash^j cf(\mathbf{Y})$ are of a different predicate symbol, and such that $\vdash$, $\Vdash^0$, ... and $\Vdash^{p(|\mathbf{s}|)}$ makes up all the predicate symbols mentioned in the head of rules in $\Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})}$. Here, we note that the size of $\Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})}$ is $O(p(|\mathbf{s}|)^3)$ and thus, is still polynomial with respect to $p(|\mathbf{s}|)$. Here, we use different predicates of those "$\Vdash^i$ " relations because we want to take the recursion "out of the equation" (so to speak). For simplicity, in the following explanations, we omit the superscripts "i" of these symbols "$\Vdash^i$" and assume it is clear from the context.

What the rules in (90) and (91) in the program $\Pi^{\text{EDGES}}_{M(\mathbf{s})}$ does is to establish the initial connections between any two configurations based on the input state transitions from $M(\mathbf{s})$, which we call *edges*. For instance, suppose $M(\mathbf{s})$ accepts string $\mathbf{s}$ in $2^{p(|\mathbf{s}|)}$ steps through the sequence of configuration changes: $cf_0, cf_1, \cdots, cf_{2^{p(|\mathbf{s}|)}-1}$. Then $\Pi^{\text{EDGES}}_{M(\mathbf{s})}$ will establish edges $cf_0 \vdash cf_1$, $cf_1 \vdash cf_2$, $\cdots$, $cf_{2^{p(|\mathbf{s}|)}-2} \vdash cf_{2^{p(|\mathbf{s}|)}-1}$. Then the transitive closure of $\Vdash$, which is defined based on $\vdash$ through transitive rules (92) in $\Pi^{\text{TRANS}}_{M(\mathbf{s})}$, is derived by the following manner: firstly, the reachability between any two configurations within two steps is derived: $cf_0 \Vdash cf_2$, $cf_1 \Vdash cf_3$, $cf_2 \Vdash cf_4$, $\cdots$, $cf_{2^{p(|\mathbf{s}|)}-3} \Vdash cf_{2^{p(|\mathbf{s}|)}-1}$, then in the second run of the evaluation, the reachability between any two configurations within four steps are derived: $cf_0 \Vdash cf_4$, $cf_1 \Vdash cf_5$, $\cdots$, $cf_{2^{p(|\mathbf{s}|)}-5} \Vdash cf_{2^{p(|\mathbf{s}|)}-1}$. This process continues until the reachability from cf_0 to $cf_{2^{p(|\mathbf{s}|)}-1}$, i.e., $cf_0 \Vdash cf_{2^{p(|\mathbf{s}|)}-1}$, is derived . As we will prove in Lemma 1, $cf_0 \Vdash cf_{2^{p(|\mathbf{s}|)}-1}$ will be derived within polynomial steps iff $M(\mathbf{s})$ accepts $\mathbf{s}$ in $2^{p(|\mathbf{s}|)}$ steps.

Before moving on to the definition of the other remaining programs, the following Lemma 6 shows that the program $\Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})}$ indeed captures the tape stings configurations of each computation steps of $M(\mathbf{s})$.

Lemma 6. *Let $s^k(\mathbf{x}, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ such that $|\mathsf{str}(\mathbf{x})| \geq 2$. Then there exists some $s^l(\mathbf{y} \circ a, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ such that $\mathbf{x} \cong \mathbf{y} \circ a \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ and $l \leq k$.*

Proof. We prove this by induction from $i = 0$ to $i = p(|\mathbf{s}|)$ such that $2^i < |\mathsf{str}(\mathbf{x})| \leq 2^{i+1}$. Indeed, for the base case, i.e., when $i = 0$ s.t. $1 < |\mathsf{str}(\mathbf{x})| \leq 2$, it is obvious the claim already holds since $\mathbf{x}$ will be of the form "$a \circ b$ " for some a, $b \in \Gamma$.

Now for our inductive step, assume that for all $j \in \{1, \ldots, i\}$, we have that $s^k(\mathbf{x}, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$, where $2^j < |\mathsf{str}(\mathbf{x})| \leq 2^{j+1}$, implies that there exists $s^l(\mathbf{y} \circ a, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ such that $\mathbf{x} \cong \mathbf{y} \circ a \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ and $l \leq k$. Then assume $s^k(\mathbf{x}, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ such that $2^{i+1} < |\mathsf{str}(\mathbf{x})| \leq 2^{i+2}$. Then there exists some $s^{k_1}(\mathbf{x}_1, \overline{\mathbf{n}}_1, \overline{\mathbf{m}})$, $s^{k_1}(\mathbf{x}_2, \overline{\mathbf{m}+\mathbf{1}}, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$, for some number $m \in \{n_1 + 1, \ldots, n_2 - 1\}$, and $k_1 < k$ and $k_2 < k$, and where $\mathbf{x} = \mathbf{x}_1 \circ \mathbf{x}_2$. In particular, since $\mathbf{x} = \mathbf{x}_1 \circ \mathbf{x}_2$ implies that $2 < |\mathbf{x}_1|$, $|\mathbf{x}_2| \leq 2^i$, then we have from the ind. hyp. that there exists some

$s^l(\mathbf{y}\circ a, \overline{\mathbf{m}+\mathbf{1}}, \overline{\mathbf{n}}_2) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ such that $\mathbf{x}_2 \cong \mathbf{y}\circ a \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ and $l \leq k_2$. Then we have from the "concatenation rules" (87) that there exists $s^{l'}\big(\mathbf{x}_1\circ(\mathbf{y}\circ a), \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2\big) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ and $l' \leq k$ as well. Moreover, by the "associativity rules" (86), we also get that there exists $s^{l'}\big((\mathbf{x}_1\circ\mathbf{y})\circ a, \overline{\mathbf{n}}_1, \overline{\mathbf{n}}_2\big) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$. Therefore, since by the rules (88) we also get that $\mathbf{x}_1\circ\mathbf{x}_2 \cong \mathbf{x}_1\circ(\mathbf{y}\circ a) \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ $\big($since $\mathbf{x}_1 \cong \mathbf{x}_1$, $\mathbf{x}_2 \cong \mathbf{y}\circ a \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$ $\big)$ and $\mathbf{x}_1\circ(\mathbf{y}\circ a) \cong (\mathbf{x}_1\circ\mathbf{y})\circ a$, then it follows that $\mathbf{x} \cong \mathbf{z}\circ a \in \mathcal{L}^{\mathbb{P}(\Pi_{M(\mathbf{s})})}(\Pi_{M(\mathbf{s})})^+$, where $\mathbf{z} = \mathbf{x}_1\circ\mathbf{y}$. □

We further note that using similar arguments in the proof in Lemma 6 shows the case $\mathbf{x} \cong a\circ\mathbf{y}$ (i.e., as opposed to case "$\mathbf{x} \cong \mathbf{y}\circ a$" that we have considered).

$$\begin{aligned}\Pi^{\text{ACCEPT}}_{M(\mathbf{s})} = \\ \big\{\, accept \leftarrow\ & cf\big(\overline{\mathbf{0}}, q_0, \rhd\circ a_0, \overline{\mathbf{0}}, \overline{\mathbf{1}} \,\|\, a_1\circ X, \overline{\mathbf{2}}, \overline{\mathbf{N}}\big) \vdash^i cf\big(\mathbf{X}_t, q, Y, \overline{\mathbf{0}}, \mathbf{X}_{tp} \,\|\, Z, \mathbf{Y}_{tp}, \overline{\mathbf{N}}\big), \\ & \overline{\mathbf{0}} < \mathbf{X}_t,\ \mathbf{X}_{tp} \prec \mathbf{Y}_{tp}, \\ & s_0\big(\rhd\circ a_0, \overline{\mathbf{0}}, \overline{\mathbf{1}}\big), s_0\big(a_1\circ X, \overline{\mathbf{2}}, \overline{\mathbf{N}}\big),\ s(Y, \overline{\mathbf{0}}, \mathbf{X}_{tp}), s(Z, \mathbf{Y}_{tp}, \overline{\mathbf{N}}) \\ \mid\ & i \in \{0,\ldots,p(|\mathbf{s}|)\},\ q \in F_{accept},\ N = |\Gamma|^{p(|\mathbf{s}|)},\ \text{and "}\rhd\text{" the left-end marker}\big\}. \end{aligned} \tag{95}$$

Then finally, we further define the program $\Pi^{\text{UNBOUND}}_{M(\mathbf{s})}$ as follows:

$$\Pi^{\text{UNBOUND}}_{M(\mathbf{s})} = \big\{\, p\big(f(V)\big) \leftarrow p(V),\ accept \,\big\}. \tag{96}$$

The program $\Pi^{\text{UNBOUND}}_{M(\mathbf{s})}$ simply allows recursion about the propagation of infinite terms of the function "f" through the predicate "p" in (96) if a configuration at an accepting state in $F_{accept} \subseteq Q$ can be reached from the initial configuration under the input string represented via predicate $\mathbf{s}_0$; otherwise, program $\Pi^{\text{UNBOUND}}_{M(\mathbf{s})}$ will make the entire program $\Pi_{M(\mathbf{s})}$ unbounded.

Then finally, we now prove the following lemma:

Lemma 7. *M accepts $\mathbf{s}$ iff* $\text{DEP}_{p[1]}\Big(\mathcal{U}^{\,2\cdot\mathbb{P}(\Pi_{M(\mathbf{s})})}\big(\Pi_{M(\mathbf{s})}, \emptyset\big)\Big) < \text{DEP}_{p[1]}\Big(\mathcal{U}^{\,3\cdot\mathbb{P}(\Pi_{M(\mathbf{s})})}\big(\Pi_{M(\mathbf{s})}, \emptyset\big)\Big)$. *Moreover, we also have that* $\mathbb{P}(\Pi_{M(\mathbf{s})}) = O\big(p(|\mathbf{s}|^{20})\big)$.

Proof. ("$\Longrightarrow$") Then we have from the definition of M that it accepts $\mathbf{s}$ within $2^{p(|\mathbf{s}|)}$ for some polynomial $p(n)$. Then by the construction of $\Pi_{M(\mathbf{s})} = \Pi^{\text{ORD}}_{M(\mathbf{s})} \cup \Pi^{\text{STR}}_{M(\mathbf{s})} \cup \Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})} \cup \Pi^{\text{ACCEPT}}_{M(\mathbf{s})} \cup \Pi^{\text{UNBOUND}}_{M(\mathbf{s})}$ and the rules in $\Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})}$ (see (90), (91) and (92)), and since by the definition of the number $\mathbb{P}(\Pi_{M(\mathbf{s})})$ (see (35)) we have that $\mathbb{P}(\Pi_{M(\mathbf{s})}) = O(p(|\mathbf{s}|^{10}))$, then it follows that the body atom

$$"\, cf\big(\overline{\mathbf{0}}, q_0, \rhd\circ a_0, \overline{\mathbf{0}}, \overline{\mathbf{1}} \,\|\, a_1\circ X, \overline{\mathbf{2}}, \overline{\mathbf{N}}\big) \vdash^i cf\big(\mathbf{X}_t, q, Y, \overline{\mathbf{0}}, \mathbf{X}_{tp} \,\|\, Z, \mathbf{Y}_{tp}, \overline{\mathbf{N}}\big)\,"$$

(for some $i \in \big\{1,\ldots,p(|\mathbf{s}|)\big\}$) will be derived so that the rule (95) will derive the (propositional) atom *accept* and will enforce the infinite propagation of terms in the recursive atom "$p\big(f(V)\big)$" in (96) since this will "fire" the rule in (96) in the program $\Pi^{\text{UNBOUND}}_{M(\mathbf{s})}$. Thus, because the only unlimited argument within the whole program $\Pi_{M(\mathbf{s})}$ is $p[1] \in \text{ARG}(\Pi)$ (where

the predicate "p" is only mentioned in the head of rule (96)), then such an acceptance of the string $\mathbf{s}$ by machine M will make the program unbounded.

("$\Longleftarrow$") If Conditions 1 and 2 holds in the Lemma 7's statement, then since the only unlimited argument of $\Pi_{M(\mathbf{s})}$ is $p[1]$ (again, please see rule (96)), then it follows from the rules in $\Pi^{\text{EDGES}}_{M(\mathbf{s})} \cup \Pi^{\text{TRANS}}_{M(\mathbf{s})}$ $\big($see (90), (91) and (92)$\big)$ and because of the atom

$$\text{“}cf\big(\overline{\mathbf{0}}, q_0, \rhd \circ a_0(\rhd), \overline{\mathbf{0}}, \overline{\mathbf{1}} \parallel a_1(\rhd) \circ X, \overline{\mathbf{2}}, \overline{\mathbf{N}}\big) \Vdash^i cf\big(\mathbf{X}_t, q, Y, \overline{\mathbf{0}}, \mathbf{X}_{tp} \parallel Z, \mathbf{Y}_{tp}, \overline{\mathbf{N}}\big)\text{”}$$

(for some $i \in \big\{1, \ldots, p(|\mathbf{s}|)\big\}$) in the body of the rule (95) (which derives the atom "*accept*"), and by the construction of the rules in $\Pi^{\text{EDGES}}_{M(\mathbf{s})}$ (see (90) and (91)), that there is a sequence of configurations: $cf_0 \vdash cf_1$, $cf_1 \vdash cf_2$, $\ldots$ $cf_{k-1} \vdash cf_k$ $(1 \leq k \leq 2^{p(|\mathbf{s}|)})$ such that the k-th configuration terminates in some accepting state of the machine M via the construction of (95). □

This completes the proof of Theorem 6. □

B.8 Proof of Theorem 7

Theorem 7. *For $k \geq 0$, deciding whether a program Π is k-EXP-bounded is $(k+1)$-EXPTIME-complete.*

Proof. (*Membership*) With the number $\mathsf{exp}_\Pi(k)$ as defined near the beginning of Section 5, we have that the $\mathsf{exp}_\Pi(k)$-steps iterations as described in Definitions 2 and 3 can be done in time:

$$\begin{aligned} O(\overbrace{\mathsf{exp}_\Pi(k)^{\,\mathsf{exp}_\Pi(k)} \cdot \ldots \cdot \mathsf{exp}_\Pi(k)^{\,\mathsf{exp}_\Pi(k)}}^{\mathsf{exp}_\Pi(k)\text{-times}}) &= O(\mathsf{exp}_\Pi(k)^{\,\mathsf{exp}_\Pi(k)\cdot\mathsf{exp}_\Pi(k)}) \\ &= O(\mathsf{exp}_\Pi(k)^{\,\mathsf{exp}_\Pi(k)^2}). \end{aligned}$$

Therefore, since:

$$\begin{aligned} & O(\mathsf{exp}_\Pi(k)^{\,\mathsf{exp}_\Pi(k)^2}) \;=\; O(2^k) \\ \Longrightarrow\; & O\Big(\log\big(\mathsf{exp}_\Pi(k)^{\,\mathsf{exp}_\Pi(k)^2}\big)\Big) \;=\; O\big(\log(2^k)\big) \\ \Longrightarrow\; & O\Big(\mathsf{exp}_\Pi(k)^2 \cdot \log\big(\mathsf{exp}_\Pi(k)\big)\Big) \;=\; O(k) \\ \Longrightarrow\; & O\Big(\mathsf{exp}_\Pi(k)^2 \cdot \mathsf{exp}_\Pi(k-1)\Big) \;=\; O(k) \\ \Longrightarrow\; & O(k) \;\leq\; O\big(\mathsf{exp}_\Pi(k)^3\big), \end{aligned}$$

then it follows that deciding if Π is k-EXP-bounded can be done in time $O\big(2^{\,\mathsf{exp}_\Pi(k)^3}\big) = O\big(\mathsf{exp}_\Pi(k+1)\big)$.

(*Hardness*) To prove the hardness direction, we first introduce the following notions. Given some number $k \geq 0$ and polynomial $p(n)$, we define the number $\mathsf{exp}_{p(n)}(i)$ inductively

as follows: (1) $\mathsf{exp}_{p(n)}(0) = p(n)$; and (2) $\mathsf{exp}_{p(n)}(i+1) = 2^{\mathsf{exp}_{p(n)}(i)}$, for $i \geq 1$. Intuitively, $\mathsf{exp}_{p(n)}(k)$ is similar to the number $\mathsf{exp}_{\Pi}(k)$ in that $p(n) = \mathbb{P}(\Pi)$ for the case of $\mathsf{exp}_{\Pi}(k)$, i.e., $p(n)$ is the constant $\mathbb{P}(\Pi)$.

Similarly to the hardness proof of Theorem 6, we reduce some arbitrary decision problem in $(k+1)$-EXPTIME to the problem of determining if a logic program is k-EXP-bounded. Before we proceed to the actual definition of these programs, we note that the key techniques we used for the hardness proof of Theorem 6 will not work for the $(k+1)$-EXPTIME case because we cannot enumerate the computation steps/tape positions using relations of tuples with polynomial arity $\big($e.g., as in the predicates "$\prec$, " "$\leq$" and "$<$" we had used in rules (70)-(79)$\big)$. As such, our key technique uses the "$\circ$" string composition to encode the linear ordering via binary strings of arbitrary length. Another problem with using techniques similar to the hardness proof of Theorem 6 is that, since we now allow the bound of steps to be some arbitrary number (i.e., $\mathsf{exp}_{p(n)}(k)$-steps rather than just the fixed polynomial $p(n)$), then it seems impossible to reduce some arbitrary $(k+1)$-EXPTIME decision problem in terms of the "boundedness" of logic programs since such a reduction can only be done in polynomial time (and thus, of polynomial size). So for this reason, rather than deriving the "positive facts" from the lower bound $\mathcal{L}^i(\Pi)$ through rules, we instead derive positive facts via encoding the rules as constraints. For instance, rather than (say) expressing some rule as

$$num(X \circ Y) \leftarrow num(X),\, num(Y),$$

we instead express it as a constraint

$$\bot \leftarrow \mathsf{not}\ num(X \circ Y),\, num(X),\, num(Y),$$

Thus, via this technique, we can control the upper bound $\mathcal{U}^i(\Pi, S)$ to be bounded since recursion only occurs via the positive propagation of terms in the lower bound $\mathcal{L}^i(\Pi)$ through constraints. With these things in mind, we now proceed to the reduction as follows.

Assuming L to be an arbitrary decision problem in $(k+1)$-EXPTIME, then from the definition of complexity class $(k+1)$-EXPTIME, there exists some *deterministic Turing machine* M such that for any string $\mathbf{s}$, $\mathbf{s} \in L$ iff M accepts $\mathbf{s}$ in at most $2^{\mathsf{exp}_{p(|\mathbf{s}|)}(k)}$-steps for some polynomial $p(n)$. Assume the Turing machine M to be the tuple $\langle Q, \Gamma, \square, \Sigma, \delta, q_0, F\rangle$ such that: (1) $Q \neq \emptyset$ is a finite set of states; (2) $\Gamma \neq \emptyset$ is a finite set of alphabet symbols; (3) $\square \in \Gamma$ is the "blank" symbol; (4) $\triangleright \in \Gamma$ is the "left-end-marker" symbol; (5) $\Sigma \subseteq \Gamma \setminus \{\square, \triangleright\}$ is the set of input symbols; (6) $\delta : (Q \setminus F) \times \big(\Gamma \setminus \{\triangleright\}\big) \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function; (7) $q_0 \in Q$ is the initial state; and lastly, (8) $F = (F_{accept} \cup F_{reject}) \subseteq Q$ is the set of final/accepting states such that $F_{accept} \cap F_{reject} = \emptyset$ and F_{accept} (F_{reject}) is the accepting (rejecting) states.

Then as promised, given a string $\mathbf{s} = a_0 \ldots a_{|\mathbf{s}|-1}$ such that $a_i \in \Sigma$ for $0 \leq i < |\mathbf{s}|$, we now construct the program $\Sigma_{M(\mathbf{s})} = \Sigma^{\text{ORD}}_{M(\mathbf{s})} \cup \Sigma^{\text{STR}}_{M(\mathbf{s})} \cup \Sigma^{\text{EDGES}}_{M(\mathbf{s})} \cup \Sigma^{\text{TRANS}}_{M(\mathbf{s})} \cup \Sigma^{\text{ACCEPT}}_{M(\mathbf{s})} \cup \Sigma^{\text{BOUND}}_{M(\mathbf{s})} \cup \Sigma^{\text{PAD}}_{M(\mathbf{s})}$. In particular, we note here that to differentiate these programs from the ones in the hardness proof of Theorem 6, we use "Σ" rather than "Π."

$$\Sigma^{\text{ORD}}_{M(\mathbf{s})} = \tag{97}$$

$$\begin{aligned}
&\big\{\, \bot \leftarrow \mathsf{not}\ num_0(\overline{0}),\\
&\quad \bot \leftarrow \mathsf{not}\ num_1(\overline{1}),\\
&\quad \bot \leftarrow \mathsf{not}\ num(\overline{0}),\\
&\quad \bot \leftarrow \mathsf{not}\ num(\overline{1}),\\
&\quad \bot \leftarrow \mathsf{not}\ num_0(X \circ Y),\ num_0(X),\ num_0(Y),\\
&\quad \bot \leftarrow \mathsf{not}\ num_1(X \circ Y),\ num_1(X),\ num_1(Y),\\
&\quad \bot \leftarrow \mathsf{not}\ num(X \circ Y),\ num(X),\ num(Y)\ \big\}\ \cup
\end{aligned} \tag{98}$$

$$\begin{aligned}
&\big\{\, \bot \leftarrow \mathsf{not}\ |\overline{0}| = |\overline{0}|,\\
&\quad \bot \leftarrow \mathsf{not}\ |\overline{0}| = |\overline{1}|,\\
&\quad \bot \leftarrow \mathsf{not}\ |\overline{1}| = |\overline{0}|,\\
&\quad \bot \leftarrow \mathsf{not}\ |\overline{1}| = |\overline{1}|,\\
&\quad \bot \leftarrow \mathsf{not}\ |X \circ Y| = |X' \circ Y'|,\ num(X \circ Y),\ num(X' \circ Y'),\\
&\qquad\quad |X| = |X'|,\ |Y| = |Y'|\ \big\}\ \cup
\end{aligned} \tag{99}$$

$$\begin{aligned}
&\big\{\, \bot \leftarrow \mathsf{not}\ X \equiv X,\ num(X),\\
&\quad \bot \leftarrow \mathsf{not}\ X \circ Y \equiv X' \circ Y',\ num(X \circ Y),\ num(X' \circ Y'),\ X \equiv Y,\ X' \equiv Y',\\
&\quad \bot \leftarrow \mathsf{not}\ X \circ (Y \circ Z) \equiv (X' \circ Y') \circ Z',\\
&\qquad\quad num\big(X \circ (Y \circ Z)\big),\ num\big((X' \circ Y') \circ Z'\big),\ X \equiv X',\ Y \equiv Y',\ Z \equiv Z'\ \big\}\ \cup
\end{aligned} \tag{100}$$

$$\begin{aligned}
&\big\{\, \bot \leftarrow \mathsf{not}\ X \circ \overline{0} \prec X' \circ \overline{1},\ num(X \circ \overline{0}),\ num(X' \circ \overline{1}),\ X \equiv X',\\
&\quad \bot \leftarrow \mathsf{not}\ V \prec W,\ num(V),\ num(W),\ V \equiv (X \circ \overline{0}) \circ Y,\ W \equiv (X' \circ \overline{1}) \circ Y',\\
&\qquad\quad X \equiv X',\ |Y| = |Y'|,\ num_1(Y),\ num_0(Y'),\\
&\quad \bot \leftarrow \mathsf{not}\ X' \circ X \prec Y,\ num_0(X'),\ num(X' \circ X),\ num(Y),\ X \prec Y,\\
&\quad \bot \leftarrow \mathsf{not}\ X \prec X' \circ Y,\ num_0(X'),\ num(X' \circ X),\ num(Y),\ X \prec Y\ \big\}\ \cup
\end{aligned} \tag{101}$$

$$\begin{aligned}
&\big\{\, \bot \leftarrow \mathsf{not}\ X < Y,\ X \prec Y,\\
&\quad \bot \leftarrow \mathsf{not}\ X < Z,\ X < Y,\ Y < Z,\\
&\quad \bot \leftarrow \mathsf{not}\ X \leq X,\ X \equiv X,\\
&\quad \bot \leftarrow \mathsf{not}\ X \leq Y,\ X < Y\ \big\},
\end{aligned} \tag{102}$$

where:

- Each of the symbols "num_i" $\big($for $j \in \{0,1\}\big)$ and "$num,$ " as mentioned in the set (98) are different predicates. In particular, the predicate "num_0" $\big(num_1\big)$ is to encode the binary strings made up of all "$\overline{0}$" $\big($"$\overline{1}$"$\big)$. As will be seen in the rules in (101), the predicate num_0 $\big(num_1\big)$ will allow us to encode the successor relation "$\prec$" of the numbers corresponding to the binary strings of the relations in predicate "num" $\big($see (101)$\big)$;
- The constants "$\overline{0}$" and "$\overline{1}$" stands for the binary digits 0 and 1, respectively;

- The expression "$|X| = |Y|$, " as defined inductively in the set (99), encodes that the strings "X" and "Y" are of the same lengths;

- The expression "$X \equiv Y$, " as defined inductively in the set (100), encodes that the binary number represented by the (binary) strings "X" and "Y" are equivalent. Intuitively, as can be seen in (100), this is equivalent to encoding that the binary strings as composed under the "$\circ$" function satisfies the associative property;

- Then lastly, the expression "$X \prec Y$, " "$X < Y$" and "$X \leq Y$, " as defined inductively in the sets (101) and (102), encodes that Y is the successor of X, X is less than Y and X is less then or equal to Y, respectively. We further mention that the last two rules in (101) is to simply extend the successor relation "$\prec$" $\big($and thus, to '$X < Y$" and "$X \leq Y$" as well via (102)$\big)$ to allow comparison between the binary strings of different lengths. For instance, given that "101011101 $\prec$ 101011110" holds, then we also get that "00000000101011101 $\prec$ 000101011110" also holds as well.

Here, the program $\Sigma^{\text{ORD}}_{M(\mathbf{s})}$ made up of the rules in the sets (98)-(102) encodes a linear ordering via the binary strings as generated through the concatenation function "$\circ$" in the rules of (98). As already mentioned above, differently from our encoding of the program $\Pi^{\text{ORD}}_{M(\mathbf{s})}$ in the proof of Theorem 6, instead of deriving "positive facts" for the lower bound $\mathcal{L}^i(\Pi)$ through rule heads, we instead derive these facts through constraints. The purpose of this is to isolate the recursion of functions and predicates only in terms of these positive facts so that the upper bound $\mathcal{U}^i(\Pi, S)$ will be controlled to be bounded.

Since the $\mathsf{exp}_{p(|\mathbf{s}|)}(k)$-times application of the constraints of the form "$\bot \leftarrow$ not $num(X \circ Y)$, $num(X)$, $num(Y)$" produces binary strings of lengths 1 to $2^{\mathsf{exp}_{p(|\mathbf{s}|)}(k)}$, then it follows that the binary strings in the relation "num" can encode numbers from 1 to $2^{2^{\mathsf{exp}_{p(|\mathbf{s}|)}(k)}} = 2^{\mathsf{exp}_{p(|\mathbf{s}|)}(k+1)}$.

$\Sigma^{\text{STR}}_{M(\mathbf{s})} =$

$$\big\{\, \bot \leftarrow \mathsf{not}\ s_0(\rhd, \overline{0}, \overline{0}) \,\big\} \cup \tag{103}$$

$$\big\{\, \bot \leftarrow \mathsf{not}\ s_0(a_{i-1}, \overline{i}, \overline{i}) \ \mid\ 1 \leq i \leq |\mathbf{s}| \ \big\} \cup \tag{104}$$

$$\big\{\, \bot \leftarrow \mathsf{not}\ s_0(\Box, T, T),\ \overline{|\mathbf{s}|} < T \,\big\} \cup \tag{105}$$

$$\big\{\, \bot \leftarrow \mathsf{not}\ s_0(Z, T_1, T_4),\ s_0(X, T_1, T_2),\ s_0(Y, T_3, T_4),\ T_1 \leq T_2,\ T_2 \prec T_3,\ T_3 \leq T_4, \tag{106}$$
$$\mid Z \in \{\, (X \circ Y),\ (Y \circ X) \,\} \ \big\} \cup$$

$$\big\{\, \bot \leftarrow \mathsf{not}\ s(a, T, T),\ num(T) \ \mid\ a \in \Gamma \,\big\} \cup \tag{107}$$

$$\big\{\, \bot \leftarrow \mathsf{not}\ s(Z, T_1, T_4),\ s(X, T_1, T_2),\ s(Y, T_3, T_4),\ T_1 \leq T_2,\ T_2 \prec T_3,\ T_3 \leq T_4, \tag{108}$$
$$\mid Z \in \{\, (X \circ Y),\ (Y \circ X) \,\} \ \big\} \cup$$

$$\begin{aligned}\big\{\, &\bot \leftarrow \mathsf{not}\ s_0\big(X \circ (Y \circ Z), T_1, T_4\big),\ s_0\big((X \circ Y) \circ Z, T_1, T_4\big),\\ &\bot \leftarrow \mathsf{not}\ s_0\big((X \circ Y) \circ Z, T_1, T_4\big),\ s_0\big(X \circ (Y \circ Z), T_1, T_4\big),\\ &\bot \leftarrow \mathsf{not}\ s\big(X \circ (Y \circ Z), T_1, T_4\big),\ s\big((X \circ Y) \circ Z, T_1, T_4\big),\\ &\bot \leftarrow \mathsf{not}\ s\big((X \circ Y) \circ Z, T_1, T_4\big),\ s\big(X \circ (Y \circ Z), T_1, T_4\big) \,\big\} \cup\end{aligned} \tag{109}$$

$$\begin{aligned}\big\{\, &\bot \leftarrow \mathsf{not}\ X \cong X,\ s\big(X, T_1, T_2\big),\\ &\bot \leftarrow \mathsf{not}\ X \circ Y \cong X' \circ Y',\ s\big(X \circ Y, T_1, T_2\big),\ s\big(X' \circ Y', T_1, T_2\big),\ X \cong X',\ Y \cong Y',\\ &\bot \leftarrow \mathsf{not}\ (X \circ Y) \circ Z \cong X' \circ (Y' \circ Z'),\ s\big((X \circ Y) \circ Z, T_1, T_2\big),\ s\big(X' \circ (Y' \circ Z'), T_1, T_2\big),\\ &\qquad X \cong X',\ Y \cong Y',\ Z \cong Z' \,\big\}.\end{aligned} \tag{110}$$

Similarly as in the description of "$\Pi^{\text{STR}}_{M(\mathbf{s})}$" in (80)-(88), the predicates "s_0" and "s" mentioned in the sets (103)-(110) of $\Sigma^{\text{STR}}_{M(\mathbf{s})}$ also encodes strings of lengths 1 to $2^{\mathsf{exp}_{p(|\mathbf{s}|)}(k)}$ for $\mathsf{exp}_{p(|\mathbf{s}|)}(k)$-steps application of the rules in (109). One crucial difference about these predicates, apart from the rules (103)-(110) now being in the form of constraints, is that the tape position arguments are now of single arity since both the tape positions and time numbers are now encoded as binary strings as composed from the "$\circ$" function in (98) rather than the fixed $p(|\mathbf{s}|)$-length tuples we used in (80)-(88). In addition, we further note that predicates "s_0" and "s" now does away with the "counter" argument as we have done in (80)-(88) because it is now not necessary to restrict the application of those recursive rules in (103)-(110).

$$\Sigma^{\text{EDGES}}_{M(\mathbf{s})} =$$
$$\big\{\, \bot \leftarrow \mathsf{not}\ cf\big(X_t, q, X \circ a, \overline{0}, X_{tp} \,\|\, c \circ Y, Y_{tp}, N_{tp}\big) \vdash cf\big(Y_t, q', X' \circ c, \overline{0}, Y_{tp} \,\|\, d \circ Y', Z_{tp}, N_{tp}\big),$$
$$X_t \prec Y_t,\ X_{tp} \prec Y_{tp},\ Y_{tp} \prec Z_{tp},\ Z_{tp} \prec N_{tp},\ s(X \circ a, \overline{0}, X_{tp}),\ s(c \circ Y, Y_{tp}, N_{tp}),$$
$$s\big((X \circ b) \circ c, \overline{0}, Y_{tp}\big),\ s(Y, Z_{tp}, N_{tp}),\ X' \cong X \circ b,\ Y \cong d \circ Y'$$
$$\mid \delta(q, a) = (q', b, R) \text{ and } c, d \in \Gamma, \text{ and } N = |\Gamma|^{p(|\mathbf{s}|)} \,\big\} \cup \tag{111}$$
$$\big\{\, \bot \leftarrow \mathsf{not}\ cf\big(X_t, q, X \circ c, \overline{0}, X_{tp} \,\|\, a \circ Y, Y_{tp}, N_{tp}\big) \vdash cf\big(Y_t, q', X' \circ d, \overline{0}, Z_{tp} \,\|\, c \circ Y', X_{tp}, N_{tp}\big),$$
$$X_t \prec Y_t,\ Z_{tp} \prec X_{tp},\ X_{tp} \prec Y_{tp},\ Y_{tp} \prec N_{tp},\ s\big((X \circ c) \circ a, \overline{0}, X_{tp}\big),\ s(Y, Y_{tp}, N_{tp}),$$
$$s(X \circ c, \overline{0}, Z_{tp}),\ s(b \circ Y, X_{tp}, N_{tp}),\ X \cong X' \circ d,\ Y' \cong b \circ Y$$
$$\mid \delta(q, a) = (q', b, L) \text{ and } c, d \in \Gamma, \text{ and } N = |\Gamma|^{p(|\mathbf{s}|)} \,\big\}. \tag{112}$$

Also similarly to its "$\Pi^{\text{EDGES}}_{M(\mathbf{s})}$" counterpart we have defined in (90) and (91), the program $\Sigma^{\text{EDGES}}_{M(\mathbf{s})}$ also establishes the initial connection between the computation steps of the Turing machine M. As already mentioned in the explanation about program "$\Sigma^{\text{STR}}_{M(\mathbf{s})}$, " here, the tape position numbers and time arguments are now of single arity to house the complex term corresponding to the binary string as produced from (98). Another difference is that the right end of tape number is not set to a specific value $\big($e.g., as in "$\overline{\mathbf{N}}$" we had done for (90) and (91)$\big)$ but rather, some arbitrary number, i.e., as encoded by the variable "N_{tp}" in the rules (111) and (112) above. Note though that because we can encode all numbers from 1 to $2^{\mathsf{exp}_{p(|\mathbf{s}|)}(k+1)}$ using only $O\big(\mathsf{exp}_{p(|\mathbf{s}|)}(k)\big)$-steps via the transitive propagations of the negative facts in (98), then it follows that such an encoding would not compromise the correctness of our reduction.

Now we further define the three programs $\Sigma^{\text{TRANS}}_{M(\mathbf{s})}$, $\Sigma^{\text{ACCEPT}}_{M(\mathbf{s})}$ and $\Sigma^{\text{BOUND}}_{M(\mathbf{s})}$ as follows:

$$\Sigma^{\text{TRANS}}_{M(\mathbf{s})} = \big\{\, \bot \leftarrow \mathsf{not}\ cf(\overline{\mathbf{X}}) \Vdash cf(\overline{\mathbf{Y}}),\ cf(\overline{\mathbf{X}}) \vdash cf(\overline{\mathbf{Y}}),$$
$$\bot \leftarrow \mathsf{not}\ cf(\overline{\mathbf{X}}) \Vdash cf(\overline{\mathbf{Z}}),\ cf(\overline{\mathbf{X}}) \Vdash cf(\overline{\mathbf{Y}}),\ cf(\overline{\mathbf{Y}}) \Vdash cf(\overline{\mathbf{Z}}) \,\big\}; \tag{113}$$

$$\Sigma^{\text{ACCEPT}}_{M(\mathbf{s})} =$$
$$\big\{\, \bot \leftarrow \mathsf{not}\ accept, \tag{114}$$
$$cf\big(\overline{0}, q_0, \triangleright \circ a_0, \overline{0}, \overline{1} \,\|\, a_1 \circ X, \overline{2}, N_{tp}\big) \Vdash cf\big(X_t, q, Y, \overline{0}, X_{tp} \,\|\, Z, Y_{tp}, N_{tp}\big),$$
$$\overline{0} < X_t,\ X_{tp} \prec Y_{tp},\ s_0(\triangleright \circ a_0, \overline{0}, \overline{1}),\ s_0(a_1 \circ X, \overline{2}, N_{tp}),\ s(Y, \overline{0}, X_{tp}),\ s(Z, Y_{tp}, N_{tp})$$
$$\mid q \in F_{accept} \text{ and } \text{“}\triangleright\text{”} \text{ the left-end marker} \,\big\}; \tag{115}$$

$$\Sigma^{\text{BOUND}}_{M(\mathbf{s})} = \big\{\, r\big(f(X)\big) \vee accept \leftarrow r(X) \,\big\}. \tag{116}$$

In (114), we note that we derive the "positive" fact "*accept*" and not its "negative" counterpart so that the rule in (116) will become bounded if "*accept*" can be derived from the lower bound $\mathcal{L}^{\mathsf{exp}_{p(|\mathbf{s}|)}}(\Sigma_{M(\mathbf{s})})^{+}$.

Then finally, we have the last program $\Sigma^{\text{PAD}}_{M(\mathbf{s})}$ defined as follows:

$$\Sigma^{\text{PAD}}_{M(\mathbf{s})} = \big\{ \bot \leftarrow p_i \mid i \in \{1, \ldots, p(|\mathbf{s}|)\} \big\}. \tag{117}$$

Intuitively, the program $\Sigma^{\text{PAD}}_{M(\mathbf{s})}$ as defined in (117) simply serves as a "padding" by introducing dummy constraints "$\bot \leftarrow p_i$" $\big($for $i \in \{1, \ldots, p(|\mathbf{s}|)\}\big)$. The purpose of such padding rules is to insure that the condition $p(|\mathbf{s}|) \leq \mathbb{P}(\Sigma_{M(\mathbf{s})})$ holds. Most importantly, we note here that since $p(n)$ is a polynomial, then so will $|\Sigma_{M(\mathbf{s})}|$ relative to the input string $\mathbf{s}$.

Then finally, to finish the proof, using similar arguments to that as in Lemma 7 of the hardness proof of Theorem 6, it follows that the following lemma also holds.

Lemma 8. *M accepts $\mathbf{s}$ iff $\Sigma_{M(\mathbf{s})} = \Sigma^{\text{ORD}}_{M(\mathbf{s})} \cup \Sigma^{\text{STR}}_{M(\mathbf{s})} \cup \Sigma^{\text{EDGES}}_{M(\mathbf{s})} \cup \Sigma^{\text{TRANS}}_{M(\mathbf{s})} \cup \Sigma^{\text{ACCEPT}}_{M(\mathbf{s})} \cup \Sigma^{\text{BOUND}}_{M(\mathbf{s})} \cup \Sigma^{\text{PAD}}_{M(\mathbf{s})}$ is k-EXP-bounded.*

This completes the proof of Theorem 7. □

References

Alviano, M., Faber, W., & Leone, N. (2010). Disjunctive ASP with functions: Decidable queries and effective computation. *Theory and Practice of Logic Programming*, *10(4-6)*, 497–512.

Baselice, S., Bonatti, P., & Criscuolo, G. (2009). On finitely recursive programs. *Theory and Practice of Logic Programming*, *9(2)*, 213–238.

Bonatti, P. (2004). Reasoning with infinite stable models. *Artificial Intelligence*, *156*(1), 75–111.

Bruynooghe, M., Codish, M., Gallagher, J., Genaim, S., & Vanhoof, W. (2007). Termination analysis of logic programs through combination of type-based norms. *ACM Transactions on Programming Languages and Systems*, *29*(2).

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, *69*(1-2), 165–204.

Calautti, M., Greco, S., Molinaro, C., & Trubitsyna, I. (2015a). logic program termination analysis using atom sizes. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-2015)*, pp. 2833–2839.

Calautti, M., Greco, S., Spezzano, F., & Trubitsyna, I. (2015b). Checking termination of bottom-up evaluation of logic programs with function symbols. *Theory and Practice of Logic Programming*, *15*(6), 854–889.

Calimeri, F., Cozza, S., Ianni, G., & Leone, N. (2008). Computable functions in ASP: theory and implementation. In *Logic Programming, 24th International Conference (ICLP-2008), Udine, Italy, December 9-13 2008, Proceedings*, pp. 407–424.

Dix, J., Gottlob, G., & Marek, W. (1996). Reducing disjunctive to non-disunctive semantics by shift-operations. *Fundamenta Informaticae*, *28*(12), 87–100.

Gebser, M., Schaub, T., & Thiele, S. (2007). Gringo : A new grounder for answer set programming. In *Logic Programming and Nonmonotonic Reasoning, 9th International*

Conference, (LPNMR-2007), Tempe, AZ, USA, May 15-17, 2007, Proceedings, pp. 266–271.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of ICLP/SLP-1988*, pp. 1070–1080.

Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programming and disjunctive databases. *New Generation Computing*, *9*, 365–386.

Genaim, S., & Codish, M. (2005). Inferring termination conditions for logic programs using backwards analysis. *Theory and Practice of Logic Programming*, *5*(1-2), 75–91.

Greco, S., Molinaro, C., & Trubitsyna, I. (2013). Bounded programs: A new decidable class of logic programs with function symbols. In *Proceedings of IJCAI-2013*, pp. 926–913.

Greco, S., Spezzano, F., & Trubitsyna, I. (2012). On the termination of logic programs with function symbols. In *Technical Communications of the 28th International Conference on Logic Programming (ICLP-2012), September 4-8, 2012, Budapest, Hungary*, pp. 323–333.

Johnson, D. (1975). Finding all the elementary circuits of a directed graph. *SIAM Journal of Computing*, *4*(1), 77–84.

Lierler, Y., & Lifschitz, V. (2009). One more decidable class of finitely ground programs. In *Logic Programming, 25th International Conference (ICLP-2009), Pasadena, CA, USA, July 14-17, 2009. Proceedings*, pp. 489–493.

Marchiori, M. (1996). Proving existential termination of normal logic programs. In *Algebraic Methodology and Software Technology, 5th International Conference, (AMAST-1996), Munich, Germany, July 1-5, 1996, Proceedings*, pp. 375–390.

Nguyen, M., Giesl, J., Schneider-Kamp, P., & Schreye, D. D. (2007). Termination analysis of logic programs based on dependency graphs. In *Logic-Based Program Synthesis and Transformation, 17th International Symposium (LOPSTR-2007), Kongens Lyngby, Denmark, August 23-24, 2007, Revised Selected Papers*, pp. 8–22.

Nishida, N., & Vidal, G. (2010). Termination of narrowing via termination of rewriting. *Applied Algebra of Engineering Communication and Computing*, *21*(3), 177–225.

Ohlebusch, E. (2001). Termination of logic programs: Transformational methods revisited. *Applied Algebra of Engineering Communicational and Computing*, *12*(1/2), 73–116.

Papadimitriou, C. (1994). *Computational Complexity*. Addison Wesley.

Schneider-Kamp, ., Giesl, J., Ströder, T., Serebrenik, A., & Thiemann, R. (2010). Automated termination analysis for logic programs with cut. *Theory and Practice of Logic Programming*, *10*(4-6), 365–381.

Schneider-Kamp, P., Giesl, J., Serebrenik, A., & Thiemann, R. (2009). Automated termination proofs for logic programs by term rewriting. *ACM Transactions on Computational Logic*, *11*(1).

Schreye, D. D., & Decorte, S. (1994). Termination of logic programs: The never-ending story. *Journal of Logic Programming*, *19/20*, 199–260.

Serebrenik, A., & Schreye, D. D. (2005). On termination of meta-programs. *Theory and Practice of Logic Programming*, *5*(3), 355–390.

Syrjänen, T. (2001). Omega-restricted logic programs. In *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, (LPNMR-2001), Vienna, Austria, September 17-19, 2001, Proceedings*, pp. 267–279.

Voets, D., & Schreye, D. D. (2011). Non-termination analysis of logic programs with integer arithmetics. *Theory and Practice of Logic Programming*, *11*(4-5), 521–536.