Stable Models for Nonmonotonic Existential Rules

Despoina Magka, Markus Krötzsch, Ian Horrocks
Department of Computer Science, University of Oxford
{desmag,markus.kroetzsch,ian.horrocks}@cs.ox.ac.uk

Abstract

In this work, we consider function-free existential rules extended with nonmonotonic negation under a stable model semantics. We present new acyclicity and stratification conditions that identify a large class of rule sets having finite, unique stable models, and we show how the addition of constraints on the input facts can further extend this class. Checking these conditions is computationally feasible, and we provide tight complexity bounds. Finally, we demonstrate how these new methods allowed us to solve relevant reasoning problems over a real-world knowledge base from biochemistry using an off-the-shelf answer set programming engine.

1 Introduction

Logic-based knowledge representation (KR) languages are widely used to model complex, structured information, e.g., in biology [Gkoutos *et al.*, 2012] and chemistry [Hastings *et al.*, 2012]. Structured knowledge models, such as the *ChEBI* database and ontology of chemical compounds [de Matos *et al.*, 2010], serve as shared reference terminologies. Reasoning supports a wide range of tasks including quality assurance, modelling, data integration, and search, and can complement statistical and machine learning approaches, e.g., in classifying chemical structures [Ferreira and Couto, 2010].

Many ontologies, including ChEBI, are based on description logics (DLs); DLs are, however, severely limited in their ability to model structures that are not tree-shaped. This explains, e.g., why ChEBI does not model molecular structures in its ontology, thus excluding its main content from logical reasoning. Numerous extension of DLs, such as *description graphs* [Motik *et al.*, 2009], provide carefully restricted kinds of rule-based or graph-based modelling, but remain largely unrealised in tools and applications. Moreover, a form of *closed-world assumption* is often needed to reason about the absence of structural features, e.g., to conclude that a molecule is inorganic if it does not contain carbon. This can be naturally modelled using a nonmonotonic DL, but such DLs currently lack tool support [Motik and Rosati, 2010].

This motivates the use of (nonmonotonic) rule languages for modelling ontologies. *Existential rules*—function-free Horn rules with existential quantifiers in rule heads—have been proposed as an ontology and data integration language [Calì *et al.*, 2010; Baget *et al.*, 2011a], and can be viewed as a restricted kind of logic programs with function symbols. Recent works have considered nonmonotonic rule-based ontology languages using stratified negation [Calì *et al.*, 2009; Magka *et al.*, 2012], stable model semantics [Eiter *et al.*, 2012], and well-founded semantics [Gottlob *et al.*, 2012]. If we additionally remove the stratification requirement, then the resulting language allows for the accurate modelling of complex finite structures such as those found in ChEBI.

Unfortunately, reasoning in these formalisms is computationally challenging. If negation is stratified, then all of these semantics agree, and programs have uniquely determined stable models; this is highly desirable and easy to check, but too restrictive for many applications. Moreover, even without negation, satisfiability, fact entailment, query answering, and the existence of finite models are all undecidable; and, while many non-stratified programs also have unique stable models, this property, too, is undecidable in general. As most ontologies are concerned with finite, uniquely determined structures, these problems raise serious doubts about the use of such formalisms in ontological modelling.

We address this issue by presenting new conditions that are computationally feasible to check, and that identify a large class of programs having finite and unique stable models. These conditions are based on an analysis of whether one rule *relies* on another, in the sense that it might either be 'triggered' or 'inhibited' by the other rule's application. These relationships allow us to define *R-acyclicity* and *R-stratification*. Specifically, our contributions are as follows:

- We define *R-acyclic* and *R-stratified* logic programs, and show that recognising such programs is coNP-complete.
- We show that R-acyclic programs have finite stable models, and that reasoning is coN2ExpTime-complete (NP-complete for data complexity).
- We show that R-stratified programs have unique stable models, so that reasoning becomes deterministic, and that if programs are also R-acyclic, reasoning becomes 2ExpTime-complete (P-complete for data complexity).
- We extend reliances to exploit *constraints*, and show that this strictly generalises our earlier criteria. Reasoning complexities carry over, but deciding R-acyclicity and R-stratification under constraints is complete for Π_2^p.

- We conduct a case study with ChEBI, which demonstrates that our conditions do not preclude suitable modelling, that R-stratification can be exploited to allow the DLV reasoner [Leone *et al.*, 2006] to scale to the large number of rules in our experiments, and that DLV can then be used to discover missing relationships in ChEBI.

We first introduce basic notions (Section 2) and discuss the use of nonmonotonic existential rules in ontological modelling (Section 3). Next, we define positive reliances and R-acyclicity, and establish related complexity results (Section 4). We then define negative reliances and R-stratification, and study the complexity of recognising these notions (Section 5). Reasoning with R-stratified programs is discussed separately (Section 6). Thereafter, we discuss the utility of constraints and extend all of our results accordingly (Section 7). We then present the ChEBI case study (Section 8), discuss related works (Section 9), and conclude (Section 10).

2 Preliminaries

We consider a standard first-order language. We use the letters a, b for constants, f, g for functions, x, y, z, u, v, w for variables, and t for terms. Lists of terms $\langle t_1,\dots,t_n\rangle$ are abbreviated as $\mathbf{t}$, similarly for lists of variables $\mathbf{x}$. We treat lists as sets when order is irrelevant. A special nullary predicate symbol $\bot$ is used to denote falsity. We use $\mathsf{Pred}(\varepsilon)$, $\mathsf{Var}(\varepsilon)$, $\mathsf{Const}(\varepsilon)$, and $\mathsf{Terms}(\varepsilon)$ to denote the predicates, variables, constants, and terms, respectively, that occur in an expression ε. Atoms, i.e., formulae without operators, are written α, β, γ. When used like a formula, sets of atoms always denote the conjunction of their members. Nonmonotonic negation is denoted **not**. For a set A of atoms, we define $\mathbf{not}\, A := \{\mathbf{not}\,\alpha \mid \alpha \in A\}$. A *nonmonotonic existential rule* (or simply *rule*) is of the form

$$r: \quad \forall \mathbf{x}.\forall \mathbf{z}.\, B^+ \wedge \mathbf{not}\, B^- \rightarrow \exists \mathbf{y}.H \tag{1}$$

where the *positive body* B^+, *negative body* B^-, and *head* H are sets (or conjunctions) of atoms without function symbols, such that $\mathsf{Var}(B^+) = \mathbf{x} \cup \mathbf{z}$, $\mathsf{Var}(B^-) \subseteq \mathbf{x} \cup \mathbf{z}$, and $\mathsf{Var}(H) \subseteq \mathbf{x} \cup \mathbf{y}$. We abbreviate r as (B^+, B^-, H). When writing rules as in (1), universal quantifiers are usually omitted. Sets of rules are called *(logic) programs*.

The *skolemisation* $\mathsf{sk}(r)$ of a rule r as in (1) is obtained by replacing each variable $y \in \mathbf{y}$ in H by a *skolem term* $f_y(\mathbf{x})$, where f_y is a fresh *skolem function symbol* of arity $|\mathbf{x}|$. Given a program P, we set $\mathsf{sk}(P) := \{\mathsf{sk}(r) \mid r \in P\}$. Assuming a fixed choice of skolem functions, sk is a bijection between rules and their skolemisations, which allows us to use the term *rule* liberally without risk of confusion. Our results refer to rules (or their skolemisations), and do not generally hold for arbitrary logic programming rules with function symbols.

A term or formula is *ground* if it contains no variables. Ground atoms are called *facts*. The *Herbrand universe* $\mathsf{HU}(P)$ of a program P is the set of all ground terms formed with constants and function symbols from $\mathsf{sk}(P)$ (using an auxiliary constant if $\mathsf{Const}(\mathsf{sk}(P)) = \emptyset$). The *grounding* $\mathsf{ground}(P)$ of P is the set of all rules that can be obtained from rules in $\mathsf{sk}(P)$ by uniformly replacing variables with terms from $\mathsf{HU}(P)$.

An *(Herbrand) interpretation* $\mathcal{M}$ is a set of facts with $\bot \notin \mathcal{M}$. Satisfaction is defined as usual: $\mathcal{M} \models B^+, \mathbf{not}\, B^-$ holds if $B^+ \subseteq \mathcal{M}$ and $B^- \cap \mathcal{M} = \emptyset$; $\mathcal{M} \models (B^+, B^-, H)$ if $\mathcal{M} \not\models B^+, \mathbf{not}\, B^-$ or $\mathcal{M} \models H$; and $\mathcal{M} \models P$ if $\mathcal{M} \models r$ for all $r \in P$. The *Gelfond-Lifschitz reduct* of P w.r.t. $\mathcal{M}$ is $\mathsf{GL}(P,\mathcal{M}) := \{(B^+, \emptyset, H) \mid (B^+, B^-, H) \in \mathsf{ground}(P) \text{ and } B^- \cap \mathcal{M} = \emptyset\}$. $\mathcal{M}$ is a *stable model* of P, written $\mathcal{M} \models_{\mathsf{SM}} P$, if $\mathcal{M} \models \mathsf{GL}(P,\mathcal{M})$ and there is no smaller model $\mathcal{M}' \subsetneq \mathcal{M}$ with $\mathcal{M}' \models \mathsf{GL}(P,\mathcal{M})$. We consider *cautious* entailment: for a program P and a fact α, $P \models \alpha$ if $\alpha \in \mathcal{M}$ for all stable models $\mathcal{M}$ of P. Consequences of programs can be computed with the T_P operator:

Definition 1. *Consider a program P and set of facts F. For a rule $r \in P$ with $\mathsf{sk}(r) = (B^+, B^-, H)$, define*

$$r(F) := \{H\theta \mid B^+\theta \subseteq F \text{ and } B^-\theta \cap F = \emptyset\}.$$

Moreover, let $T_P(F) := F \cup \bigcup_{r \in P} r(F)$ and define

$$T_P^0(F) := F, \quad T_P^{i+1}(F) := T_P(T_P^i(F)), \quad T_P^\infty(F) := \bigcup_{i \geq 0} T_P^i(F).$$

Given a program P, a sequence of disjoint programs $\mathbf{P} = P_1,\dots,P_n$ is a *stratification* of P if $P = \bigcup_{i=1}^n P_i$ and, for all programs $P_i, P_j \in \mathbf{P}$, rules $(B_1^+, B_1^-, H_1) \in P_i$ and $(B_2^+, B_2^-, H_2) \in P_j$, and every predicate $\mathsf{R} \in \mathsf{Pred}(H_1)$, we have: (i) if $\mathsf{R} \in \mathsf{Pred}(B_2^+)$ then $i \leq j$, and (ii) if $\mathsf{R} \in \mathsf{Pred}(B_2^-)$ then $i < j$. The elements of $\mathbf{P}$ are called *strata*. P is *stratified* if it has a stratification. The T_P operator can be used to characterise stable models; for stratified programs, we even obtain a deterministic computation procedure [Apt and Bol, 1994].

Fact 1. *Given a program P, a set of facts F, and a stable model $\mathcal{M} \models_{\mathsf{SM}} P \cup F$, we have $\mathcal{M} = T^\infty_{\mathsf{GL}(P,\mathcal{M})}(F)$.*

If $\mathbf{P} = P_1,\dots,P_n$ is a stratification of P, then $\mathcal{M} := T^\infty_{P_n}(\dots T^\infty_{P_1}(F)\dots)$ is the unique stable model of P if $\bot \notin \mathcal{M}$.

3 Modelling with Nonmonotonic Rules

Rule-based formalisms are well suited for modelling relational structures, irrespective of whether these structures are tree-shaped or cyclic. As a practical example, we consider the modelling of chemical compounds and their relations in bioinformatics. We model concepts found in the popular *ChEBI* database and ontology, which contains information about chemical entities and classes [de Matos *et al.*, 2010; Hastings *et al.*, 2013].

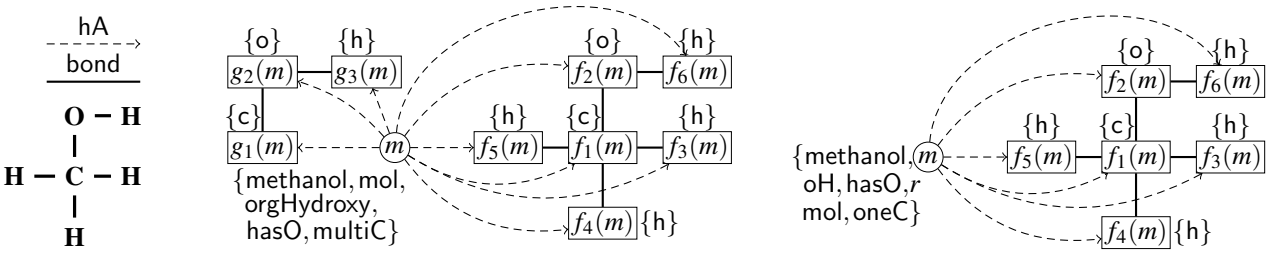
The structure of molecules can be readily represented as a logical structure. For example, the formula $M_{\mathsf{H2O}}(x,y,z) := \mathsf{o}(x) \wedge \mathsf{bond}(x,y) \wedge \mathsf{bond}(x,z) \wedge \mathsf{h}(y) \wedge \mathsf{h}(z)$ could represent a water molecule (using unidirectional bonds for simplicity). We model molecules as members of a unary predicate mol, related to their constituting atoms by the predicate hA (has atom). The following rule infers the structure of the six atoms of methanol (CH_3OH), described by the formula $M_{\mathsf{CH3OH}}(\mathbf{y})$:

$$\mathsf{methanol}(x) \rightarrow \exists \mathbf{y}.\mathsf{mol}(x) \wedge M_{\mathsf{CH3OH}}(\mathbf{y}) \wedge \textstyle\bigwedge_{i=1}^6 \mathsf{hA}(x,y_i) \tag{2}$$

Molecules can also be classified by their structure, e.g., to identify molecules that contain oxygen, or organic hydroxy molecules (those with a substructure C-O-H):

$$\mathsf{hA}(x,y) \wedge \mathsf{o}(y) \rightarrow \mathsf{hasO}(x) \tag{3}$$

$$M_{\mathsf{COH}}(\mathbf{y}) \wedge \textstyle\bigwedge_{i=1}^3 \mathsf{hA}(x,y_i) \rightarrow \mathsf{orgHydroxy}(x) \tag{4}$$



(a) Methanol molecule (b) Stable model of (2)–(7) and $\mathsf{methanol}(m)$ (c) Stable model for modified rules (8) and (9)

Figure 1: The chemical structure and the models of methanol

It is not hard to express syntactic identity with a predicate =, predefined in most rule engines. To this end, we are making the following two assumptions.

- For every constant c in the program or set of facts, the set of facts also contains $c = c$.
- For every existentially quantified variable y in the head of a rule, the head also contains the atom $y = y$.

With the above assumptions, we can treat = like any other predicate, also in our later analysis of programs. Since all facts about equality are identities, we do not need rules to axiomatise properties like symmetry or transitivity of =; indeed, such rules would have a significant influence on the properties of the program. Using **not** we can express syntactic inequality and define, e.g., molecules with exactly one carbon atom:

$$\textstyle\bigwedge_{i=1}^{2} \mathsf{hA}(x,y_i) \wedge \mathsf{c}(y_i) \wedge \mathbf{not}\ y_1 = y_2 \rightarrow \mathsf{multiC}(x) \quad (5)$$

$$\mathsf{mol}(x) \wedge \mathsf{hA}(x,y) \wedge \mathsf{c}(y) \wedge \mathbf{not}\ \mathsf{multiC}(x) \rightarrow \mathsf{oneC}(x) \quad (6)$$

The fact $\mathsf{methanol}(a)$ and the rules (2)–(6) have a unique stable model (using skolem functions $f_1, \ldots, f_6$ for (2)):

$$\mathcal{M}_1 := \{\mathsf{methanol}(a), \mathsf{hasO}(a), \mathsf{orgHydroxy}(a), \mathsf{oneC}(a), \mathsf{mol}(a), \mathsf{hA}(a, f_i(a))_{i=1}^{6}, M_{\mathsf{CH3OH}}(f_1(a), \ldots, f_6(a))\}$$

We can thus conclude, e.g., that methanol is an organic hydroxy molecule. To obtain such inferences for organic hydroxy molecules in general, we can use another rule:

$$\mathsf{orgHydroxy}(x) \rightarrow \exists \mathbf{y}.M_{\mathsf{COH}}(\mathbf{y}) \wedge \textstyle\bigwedge_{i=1}^{3} \mathsf{hA}(x,y_i) \quad (7)$$

The fact $\mathsf{orgHydroxy}(b)$ and the rules (3)–(7) have a unique stable model (using skolem functions $g_1, \ldots, g_3$ for (7)):

$$\mathcal{M}_2 := \{\mathsf{orgHydroxy}(b), \mathsf{hasO}(b), \mathsf{hA}(b, g_i(b))_{i=1}^{3}, M_{\mathsf{COH}}(g_1(b), g_2(b), g_3(b))\}$$

Hence, organic hydroxy molecules are structures with oxygen, as expected. However, if we consider all of the above rules and facts together, then rather than $\mathcal{M}_1 \cup \mathcal{M}_2$ we obtain $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \{\mathsf{hA}(a, g_i(a))_{i=1}^{3}, M_{\mathsf{COH}}(g_1(a), g_2(a), g_3(a)), \mathsf{multiC}(a)\} \setminus \{\mathsf{oneC}(a)\}$ as the unique stable model, since rule (7) is applicable to $\mathsf{orgHydroxy}(a)$. Thus, the stable model is no longer a faithful representation of the molecule a, which is wrongly classified as a multi-carbon molecule.

Nonmonotonic negation can be used to overcome this problem. We replace rules (4) and (7) by the following, where we abbreviate orgHydroxy by oH:

$$M_{\mathsf{COH}}(\mathbf{y}) \wedge \textstyle\bigwedge_{i=1}^{3} \mathsf{hA}(x,y_i) \wedge \mathbf{not}\ n(y_i) \rightarrow \mathsf{oH}(x) \wedge r(x) \quad (8)$$

$$\mathsf{oH}(x) \wedge \mathbf{not}\ r(x) \rightarrow \exists \mathbf{y}.M_{\mathsf{COH}}(\mathbf{y}) \wedge \textstyle\bigwedge_{i=1}^{3} \mathsf{hA}(x,y_i) \wedge n(y_i) \quad (9)$$

The predicates r ('recognised') and n ('new') ensure that only one of these rules is applicable to a given structure. The above facts with rules (2), (3), (5), (6), (8), and (9) have the unique stable model $\mathcal{M}_1 \cup \mathcal{M}_2 \cup \{r(a), n(g_1(b)), n(g_2(b)), n(g_3(b))\}$, as desired. However, the resulting set of rules is not stratified, which causes various problems. First, we cannot be sure that the stable model will be unique for other sets of facts. Second, rule engines may need to apply more complex algorithms to find the stable model. Our experiments in Section 8 suggest that this may cause performance issues that prevent rule engines from computing entailments at all. The goal of this work is to overcome these issues.

4 Positive Reliances and R-Acyclicity

As recalled in Fact 1, every stable model of a logic program can be obtained from a (possibly infinite) sequence of consecutive rule applications. Insights about the semantics of a program can thus be gained by analysing, for all pairs of rules r_1 and r_2, whether an application of r_1 can potentially enable a later application of r_2. In this section, we formalise this idea of *positive reliance* between rules and define R-acyclic programs, which have stable models of bounded size.

Definition 2 (Positive Reliance)**.** *Let r_1 and r_2 be rules such that $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$ and $\mathsf{sk}(r_2) = (B_2^+, B_2^-, H_2)$; w.l.o.g. assume that $\mathsf{Var}(r_1) \cap \mathsf{Var}(r_2) = \emptyset$. Rule r_2* positively relies *on r_1 (written $r_1 \xrightarrow{+} r_2$) if there exists a set of facts F that contains no skolem terms and a substitution θ such that:*

$$B_1^+\theta \subseteq F \quad \text{(P1)} \qquad B_2^-\theta \cap (F \cup H_1\theta) = \emptyset \quad \text{(P4)}$$

$$B_1^-\theta \cap F = \emptyset \quad \text{(P2)} \qquad B_2^+\theta \not\subseteq F \quad \text{(P5)}$$

$$B_2^+\theta \subseteq F \cup H_1\theta \quad \text{(P3)} \qquad H_2\theta \not\subseteq F \cup H_1\theta \quad \text{(P6)}$$

Thus, $r_1 \xrightarrow{+} r_2$ holds if there is a situation (defined by F) where r_1 is applicable (P1)/(P2), r_2 is not applicable (P5), and applying r_1 allows r_2 to derive something new (P3)/(P4)/(P6).

Example 1. *Consider rule $r_{(4)}$ of* (4)*, and rule $r'_{(7)}$ obtained from* (7) *by replacing variable x with x'. We find*

that $r_{(4)} \xrightarrow{\pm} r'_{(7)}$ *since* $F := \{M_{\mathsf{COH}}(\mathbf{b})\} \cup \{\mathsf{hA}(a,b_i)\}_{i=1}^{3}$ *and* $\theta := \{x \mapsto a, \mathbf{y} \mapsto \mathbf{b}, x' \mapsto a\}$ *satisfy* (P1)–(P6).

In contrast, $r'_{(7)} \not\xrightarrow{\pm} r_{(4)}$. *Intuitively,* $r_{(4)}$ *can only derive facts that are already necessary to apply* $r'_{(7)}$ *in the first place, thus violating* (P6). *More formally, suppose that* $r'_{(7)} \xrightarrow{\pm} r_{(4)}$ *could be shown using* F' *and* θ'. *By* (P1) *and* (P6), $\theta'(x) \neq \theta'(x')$. *Thus, by* (P3), $\mathsf{hA}(x,y_i)\theta' \in F'$ *for all* $i \in \{1,2,3\}$. *Since* F' *must not contain skolem terms,* $\theta'(y_i) \neq g_i(\theta'(x'))$, *so* $M_{\mathsf{COH}}(\mathbf{y})\theta' \subseteq F'$, *again by* (P3). *Thus* (P5) *would be violated.*

Note that Definition 2 does not consider the syntactic equality predicate = discussed in Section 3. In particular, F is not required to contain all facts $c = c$. This condition could easily be added, but even in its current form, the definition will merely lead to a very small amount of additional reliances in cases of little practical interest (namely for rules with body atoms **not** $x = x$). Since reliances are only approximating actual relationships between rules in any case, such an overestimation does not impair the correctness of any of our subsequent results.

Various previous works consider similar notions. The *activation* relation by Greco *et al.* [2012] is most similar to Definition 2, but allows F to contain function terms to accommodate arbitrary disjunctive logic programs with functions. Our stronger restriction is needed to show $r'_{(7)} \not\xrightarrow{\pm} r_{(4)}$ in Example 1. This illustrates how we can take advantage of the specific structure of existential rules to discard certain potential interactions. Other similar notions are the $\prec$ relation by Deutsch *et al.* [2008] and the *rule dependency* by Baget *et al.* [2011a], neither of which cover negation. Baget *et al.* omit condition (P6), needed to show $r'_{(7)} \not\xrightarrow{\pm} r_{(4)}$ in Example 1.

If a finite program has an infinite stable model, some rule with an existential quantifier must be applicable an infinite number of times. This, however, requires that there is a cycle in rule reliances, motivating the following definition.

Definition 3 (R-Acyclic)**.** *A program* P *is* R-acyclic *if there is no cycle of positive reliances* $r_1 \xrightarrow{\pm} \ldots \xrightarrow{\pm} r_n \xrightarrow{\pm} r_1$ *that involves a rule with an existential quantifier.*

Example 2. *The complete list of positive reliances for the rules* $r_{(2)}, \ldots, r_{(7)}$ *is* $r_{(2)} \xrightarrow{\pm} r_{(3)}$, $r_{(2)} \xrightarrow{\pm} r_{(4)}$, $r_{(2)} \xrightarrow{\pm} r_{(5)}$, $r_{(2)} \xrightarrow{\pm} r_{(6)}$, $r_{(4)} \xrightarrow{\pm} r_{(7)}$, $r_{(7)} \xrightarrow{\pm} r_{(3)}$, $r_{(7)} \xrightarrow{\pm} r_{(5)}$, *and* $r_{(7)} \xrightarrow{\pm} r_{(6)}$. *Thus the program is R-acyclic. To model* $=$, *we assume that* $y_i = y_i$ *is derived for all existential variables* y_i.

We prove that checking positive reliance for two rules is NP-complete. Similar results are shown by Deutsch *et al.* [2008] and by Baget *et al.* [2011b] for rules without negation. The complexity refers to the size of the two involved rules rather than to the size of the whole program: in practice, positive reliances can be checked efficiently by checking the applicability of one of the rules to a linear number of facts.

Theorem 1. *Given rules* r_1 *and* r_2, *the problem of deciding whether* $r_1 \xrightarrow{\pm} r_2$ *is* NP*-complete. Checking whether a program* P *is R-acyclic is* coNP*-complete.*

Proof. We first show the complexity for deciding $r_1 \xrightarrow{\pm} r_2$. For this let $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$, $\mathsf{sk}(r_2) = (B_2^+, B_2^-, H_2)$, and $\mathsf{Var}(r_1) \cap \mathsf{Var}(r_2) = \emptyset$.

Membership: Assume that $r_1 \xrightarrow{\pm} r_2$. Then there exists a set of facts F and a substitution θ that satisfy the conditions of Definition 2. By conditions (P1) and (P3) we can assume w.l.o.g. that

$$F \subseteq (B_1^+ \cup B_2^+)\theta. \tag{10}$$

Thus the size of θ is $\mathsf{Var}(B_1^+ \cup B_2^+)$ which, like the size of F, is polynomial in the size of r_1 and r_2. It is thus possible to guess F and θ, and to verify all conditions of Definition 2 in polynomial time.

Hardness: The problem of checking whether there exists a homomorphism from a set of atoms to another set of atoms is known to be NP-complete [Mugnier, 2009]. Therefore we reduce the problem of homomorphism checking to the problem of checking positive reliance. In order to do that, we assign to each instance of the homomorphism problem for two sets of atoms $Q \neq \emptyset$ and Q' a pair of rules r_1 and r_2, such that:

$$\exists \text{ homomorphism } h : Q \to Q' \quad \Leftrightarrow \quad r_1 \xrightarrow{\pm} r_2 \tag{11}$$

Let Q and Q' be two such sets of atoms with $\mathsf{Var}(Q) = \mathbf{x}$ and $\mathsf{Var}(Q') = \mathbf{x}'$. For each term s and set of atoms R, we define $\mathsf{Aster}(s,R) := \{\hat{\mathsf{P}}(s,\mathbf{t}) \mid \mathsf{P}(\mathbf{t}) \in R, |\mathbf{t}| \geq 0\}$. Rules r_1 and r_2 are defined as follows:

$$r_1 : \quad \mathsf{Start} \to \exists y', \mathbf{x}'.\mathsf{Aster}(y', Q')$$
$$r_2 : \quad \mathsf{Aster}(y, Q) \to \mathsf{Goal}$$

where Start and Goal are fresh nullary predicates, and y and y' are distinct fresh variables. Clearly, the size of r_1 and r_2 is polynomial in the size of Q' and Q. Consider a substitution $\tau := \{y' \mapsto c_{y'}\} \cup \{x' \mapsto c_{x'} \mid x' \in \mathbf{x}'\}$, where $c_{y'}$ and $\mathbf{c}_{\mathbf{x}'}$ are the constant and the vector of constants used to skolemise y' and $\mathbf{x}'$, respectively.

First, we show $\Rightarrow$ of (11). Given a homomorphism $h : Q \to Q'$, let $F := \{\mathsf{Start}\}$, let $\sigma := \{x \mapsto h(x) \mid x \in \mathbf{x}\} \cup \{y \mapsto y'\}$ and let $\theta := \sigma \circ \tau$. To show the claim, we check whether the conditions of positive reliance are satisfied:

(P1) $B_1^+\theta \subseteq F$ because $\{\mathsf{Start}\} \subseteq \{\mathsf{Start}\}$.

(P2) $B_1^-\theta \cap F = \emptyset$ because $B_1^- = \emptyset$.

(P3) $B_2^+\theta \subseteq F \cup H_1\theta$ by $\mathsf{Aster}(y,Q)\theta \subseteq \mathsf{Aster}(c_{y'}, Q'\tau)$, which is a consequence of $h(Q) \subseteq Q'$ and $\theta(y) = c_{y'}$.

(P4) $B_2^-\theta \cap (F \cup H_1\theta) = \emptyset$ because $B_2^- = \emptyset$.

(P5) $B_2^+\theta \not\subseteq F$ because $F = \{\mathsf{Start}\}$ and there is no atom with a nullary predicate in $B_2^+\theta$.

(P6) $H_2\theta \not\subseteq F \cup H_1\theta$ by $\{\mathsf{Goal}\} \not\subseteq \{\mathsf{Start}\} \cup \mathsf{Aster}(c_{y'}, Q')$.

Now we show $\Leftarrow$ of (11). If $r_1 \xrightarrow{\pm} r_2$, then there exists a set of facts F and a substitution θ such that (P1)–(P6) hold. By conditions (P3) and (P6), there exists at least one atom $\alpha \in B_2^+$ and an atom $\alpha' \in H_1$, such that $\alpha\theta = \alpha'$. Since α and α' are of the form $\hat{\mathsf{P}}(y,\mathbf{t})$ and $\hat{\mathsf{P}}(c_{y'}, \mathbf{c}_{\mathbf{t}'})$, we have $\theta(y) = c'_y$. Additionally, since y occurs in every atom of B_2^+, $\theta(y) = c'_y$ and c'_y is a skolem constant, we have $B_2^+\theta \cap F = \emptyset$. Now by $B_2^+\theta \cap F = \emptyset$ and (P3), we have $B_2^+\theta \subseteq H_1$. Thus, for every atom in B_2^+ of the form $\hat{\mathsf{P}}(y,\mathbf{t})$ there exists an atom of the form $\hat{\mathsf{P}}(c_{y'}, \mathbf{c}_{\mathbf{t}'})$ such that $\hat{\mathsf{P}}(y,\mathbf{t})\theta = \hat{\mathsf{P}}(c_{y'}, \mathbf{c}_{\mathbf{t}'})$. Since $\theta(y) = c_{y'}$,

for every atom $\mathsf{P}(\mathbf{t}) \in Q$, there exists an atom $\mathsf{P}(\mathbf{c}_{\mathbf{t}'}) \in Q'$, such that $\mathsf{P}(\mathbf{t})\theta = \mathsf{P}(\mathbf{t}')$. So, the function $h = \theta \circ \tau^{-1}$ is a homomorphism from Q to Q'.

This shows (11) and establishes the claimed hardness result.

Finally, we show that checking R-acyclicity is coNP-complete. Membership follows since it can be checked in NP that P is *not* R-acyclic. Indeed, if P is not R-acyclic, it has a cycle of length $n \leq |P|$. One can guess such a cycle $r_0 \xrightarrow{+} \ldots \xrightarrow{+} r_{n-1} \xrightarrow{+} r_0$ and justifications F_i, θ_i for each of the positive reliances $r_i \xrightarrow{+} r_{(i+1) \bmod n}$, and these choices can be verified in polynomial time. For hardness, consider a rule $r_3 : \mathsf{Goal} \rightarrow \mathsf{Start}$ and the rules r_1 and r_2 as constructed in the above hardness proof. Clearly, the program $\{r_1, r_2, r_3\}$ is R-acyclic if and only if there is no homomorphism $h : Q \rightarrow Q'$. □

The main result of this section shows that entailment under stable model semantics is decidable for R-acyclic programs. Hardness for coN2ExpTime can be shown by reducing the word problem of 2ExpTime-bounded non-deterministic Turing machines to cautious entailment, adapting constructions by Calì *et al.* [2012] and Krötzsch and Rudolph [2011].

Theorem 2. *Let P be an R-acyclic program and let $F \cup \{\alpha\}$ be a set of facts. Every stable model of $P \cup F$ has size doubly exponential in the size of P and polynomial in the size of F. Deciding $P \cup F \models \alpha$ is* coN2ExpTime*-complete w.r.t. program complexity and* coNP*-complete w.r.t. data complexity.*

The subsequent proof of this theorem involves a contradiction argument: we show that if a desired property does not hold, then there must be reliances that violate R-acyclicity. According to Definitions 2 and 4, in order to show the existence of a reliance we need to define suitable sets of facts F and θ, where F does not contain skolem terms. Facts that are obtained during a derivation do usually not have this property, but a suitable set of facts can still be obtained by replacing skolem terms with fresh constants. This construction is used in several proofs in this paper, so we give a formal definition.

Notation 1. *For a set of facts F, the mapping γ_F on ground terms is recursively defined as follows:*

- *if t is a constant, let $\gamma_F(t) := t$;*
- *if $t \in \mathsf{Terms}(F) \setminus \mathsf{Const}(F)$, let $\gamma_F(t) := c_t$ be a unique fresh constant symbol;*
- *if $t = f(\mathbf{s}) \notin \mathsf{Terms}(F)$, let $\gamma_F(t) := f(\gamma_F(\mathbf{s}))$.*

We apply γ_F to (sets of) formulae and to substitutions by applying it to all terms in these structures.

Proof of Theorem 2. A functional term $f(\mathbf{t})$ is called *cyclic* if $\mathbf{t}$ either contains the function symbol f, or (recursively) if $\mathbf{t}$ contains a cyclic term. We first prove that the stable models of $P \cup F$ do never contain any fact that uses a cyclic term. Suppose for a contradiction that there is a stable model $\mathcal{M}$ of $P \cup F$ and a fact $\alpha \in \mathcal{M}$, such that α contains a cyclic term. By Fact 1, we have $\mathcal{M} = T^{\infty}_{\mathsf{GL}(P,\mathcal{M})}(F)$. Thus, α is derived by some finite chain of applications of rules from $\mathsf{GL}(P,\mathcal{M})$ to F. Let $r_1, \ldots, r_m$ be a minimal sequence of rules $r_i \in \mathsf{GL}(P,\mathcal{M})$ that derive α, that is, $\alpha \in r_m(\ldots r_1(F)\ldots)$ and this property does not hold if any of the rules are omitted from the sequence. Let $F_1 := F$ and let $F_{i+1} := r_i(F_i)$ for all $i \in \{1, \ldots, m-1\}$. Let $r'_i \in P$ be a rule and θ_i be a ground substitution such that $r_i \in \mathsf{GL}(P,\mathcal{M})$ is obtained from $r'_i\theta_i$ by removing negative body atoms that do not occur in $\mathcal{M}$. We show that, for every $i \in \{1, \ldots, m-1\}$, there is a reliance $r'_i \xrightarrow{+} r'_{i+1}$. Let $\mathsf{sk}(r'_i) = (B^+_i, B^-_i, H_i)$ and $\mathsf{sk}(r'_{i+1}) = (B^+_{i+1}, B^-_{i+1}, H_{i+1})$, and set $G := \gamma_{F_i}(F_i)$ and $\sigma := \gamma_{F_i}(\theta_i \cup \theta_{i+1})$ (it might be necessary to rename variables in r'_{i+1} to ensure $\mathsf{Var}(r'_i) \cap \mathsf{Var}(r'_{i+1}) = \emptyset$). We show that the conditions of Definition 2 are satisfied.

(P1) $B^+_i\sigma \subseteq G$ since $r_i = \mathsf{sk}(r'_i)\theta_i$ is applicable to F_i.

(P2) $B^-_i\sigma \cap G = \emptyset$ since $r_i = \mathsf{sk}(r'_i)\theta_i \in \mathsf{GL}(P,\mathcal{M})$.

(P3) $B^+_{i+1}\sigma \subseteq G \cup H_i\sigma$ since $G \cup H_i\sigma = \gamma_{F_i}(F_{i+1})$.

(P4) $B^-_{i+1}\sigma \cap (G \cup H_i\sigma) = \emptyset$ since $r_{i+1} = \mathsf{sk}(r'_{i+1})\theta_{i+1} \in \mathsf{GL}(P,\mathcal{M})$.

(P5) $B^+_{i+1}\sigma \not\subseteq G$ since otherwise r_i would not be required to derive α.

(P6) $H_{i+1}\sigma \not\subseteq G \cup H_i\sigma$ since otherwise r_{i+1} would not be required to derive α.

Thus, we obtain a chain of reliances $r'_1 \xrightarrow{+} \ldots \xrightarrow{+} r'_m$. Since F does not contain functional terms of the form $g(\mathbf{s})$, every such (sub)term that occurs in α must have been introduced by the application of some rule r_i. There is only exactly one rule $r_g \in P$ that can introduce a term of form $g(\mathbf{s})$. Thus, if α contains a cyclic term that contains a subterm $f(\mathbf{t})$ such that $\mathbf{t}$ contains f, then there are two distinct indices $k < \ell$ such that $r'_k = r'_\ell = r_f$. The reliances shown above thus yield a cycle $r'_k \xrightarrow{+} \ldots \xrightarrow{+} r'_{\ell-1} \xrightarrow{+} r'_\ell = r'_k$. Since r_f contains an existential quantifier, this shows that P is not R-acyclic, contradicting our assumptions.

Now we establish an upper bound on the number of distinct terms that a stable model of $P \cup F$ may contain. For the computation of the bound, let c be the number of constants in $F \cup P$, let e be the maximal number of variables in any rule of P, assume that $\mathsf{sk}(P)$ contains d distinct function symbols and let $a > 1$ be some upper bound for the arity of function symbols. Each term can be represented by a tree where the outermost function symbol is the root, nested function symbols are the inner nodes and constants are the leaves. For terms that are not cyclic, this tree has depth at most $d+1$, with at most a^d inner nodes and at most a^d leaves. Each inner node is marked by one of the d function symbols, and each leaf is marked by one of the c constants, so there are at most $c^{a^d} \cdot d^{a^d} = (c \cdot d)^{a^d}$ distinct terms that occur in any stable model of $P \cup F$.

Grounding P with $(c \cdot d)^{a^d}$ terms yields a maximal number of $((c \cdot d)^{a^d})^e = (c \cdot d)^{e \cdot a^d}$ propositional rules. The entailments of P agree with the entailments of this propositional program. Cautious entailment for propositional logic programs with negation can be decided in coNP [Dantsin *et al.*, 2001], so entailment over P can be decided in coN2ExpTime. If the size of P is fixed, the bound on the number of grounded rules is of the form $k_1 \cdot c^{k_2}$, where k_1 and k_2 are constants. The size of the grounded program is thus polynomial in the size of F, so entailment can be decided in coNP w.r.t. data complexity.

1. **Initialisation:** if $w = \alpha_0 \dots \alpha_m$

$$\mathsf{min}_k(x_0) \wedge \bigwedge_{0 \le i \le m} \mathsf{succ}_k(x_i, x_{i+1}) \rightarrow \mathsf{state}_{q_0}(x_0) \wedge \bigwedge_{0 \le i \le m} \mathsf{symbol}_{\alpha_i}(x_0, x_i) \wedge \mathsf{symbol}_{\square}(x_0, x_{m+1})$$

$$\mathsf{min}_k(x_0) \wedge \mathsf{symbol}_{\square}(x_0, x) \wedge \mathsf{succ}_k(x, y) \rightarrow \mathsf{symbol}_{\square}(x_0, y)$$

2. **Transition rules:** for all $\delta = \langle q, \alpha, q', \alpha', m \rangle \in \Delta$ with $\Delta[\delta] := \{\varepsilon \in \Delta \mid \varepsilon = \langle q, \alpha, q_*, \alpha_*, m_* \rangle, \varepsilon \neq \delta\}$

$$\mathsf{state}_q(v) \wedge \mathsf{head}(v,x) \wedge \mathsf{symbol}_\alpha(v,x) \wedge \mathsf{succ}_k(y,x) \wedge \mathsf{succ}_k(v,v') \wedge \bigwedge_{\varepsilon \in \Delta[\delta]} \mathbf{not}\ \mathsf{select}_\varepsilon(v) \rightarrow \mathsf{state}_{q'}(v') \wedge \mathsf{head}(v',y) \wedge \mathsf{symbol}_{\alpha'}(v',x) \wedge \mathsf{select}_\delta(v) \quad \text{if } m = l$$

$$\mathsf{state}_q(v) \wedge \mathsf{head}(v,x) \wedge \mathsf{symbol}_\alpha(v,x) \wedge \mathsf{succ}_k(x,y) \wedge \mathsf{succ}_k(v,v') \wedge \bigwedge_{\varepsilon \in \Delta[\delta]} \mathbf{not}\ \mathsf{select}_\varepsilon(v) \rightarrow \mathsf{state}_{q'}(v') \wedge \mathsf{head}(v',y) \wedge \mathsf{symbol}_{\alpha'}(v',x) \wedge \mathsf{select}_\delta(v) \quad \text{if } m = r$$

3. **Inertia:** for all $\alpha \in \Sigma$

$$\mathsf{succ}_k(v,v') \wedge \mathsf{head}(v,x) \wedge \mathsf{succt}(x,y) \wedge \mathsf{symbol}_\alpha(v,y) \rightarrow \mathsf{symbol}_\alpha(v',y)$$
$$\mathsf{succ}_k(v,v') \wedge \mathsf{head}(v,x) \wedge \mathsf{succt}(y,x) \wedge \mathsf{symbol}_\alpha(v,y) \rightarrow \mathsf{symbol}_\alpha(v',y)$$

4. **Acceptance:** for each accepting state $q_a \in Q$

$$\mathsf{state}_{q_a}(v) \rightarrow \mathsf{accept}$$

Figure 2: Program $P^{\mathsf{comp}}_{\mathcal{T},w}$ simulating computation of an NTM $\mathcal{T}$ over w

The claimed coNP-hardness w.r.t. data complexity is an immediate consequence of the fact that entailment is coNP-hard w.r.t. data-complexity for Datalog with negation (i.e., rules without existential quantifiers or function symbols) [Dantsin *et al.*, 2001]. Indeed, every Datalog program with negation is clearly R-acyclic, since it does not contain existential quantifiers.

To prove coN2ExpTime-hardness w.r.t. program complexity, we show that for every non-deterministic Turing machine (NTM) $\mathcal{T} = \langle Q, \Sigma, \Delta, q_0 \rangle$, word $w \in \Sigma^*$, and number $k \ge 1$, we can construct an R-acyclic program $P_{\mathcal{T},w,k}$ with a special propositional symbol reject such that

$$\mathcal{T} \text{ accepts } w \text{ in time } 2^{2^k} \;\Leftrightarrow\; P_{\mathcal{T},w,k} \not\models \mathsf{reject}. \qquad (*)$$

The proof adapts standard reduction techniques [Bidoit and Froidevaux, 1991; Dantsin *et al.*, 2001; Calì *et al.*, 2012; Krötzsch and Rudolph, 2011].

Let $\mathcal{T} = \langle Q, \Sigma, \Delta, q_0 \rangle$ be an NTM such that the transition relation Δ consists of tuples of the form $\langle q, \alpha, q', \alpha', m \rangle$, where $q, q' \in Q$, $\alpha, \alpha' \in \Sigma$ and $m \in \{l, r\}$ depending on whether the head moves left or right.

First, given $k \ge 1$, we construct an R-acyclic program P^{dexp}_k of polynomial size, which encodes a chain of 2^{2^k} elements. We use this double-exponentially long chain to model both the tape of cells and the timeline of instants. Our construction follows Calì *et al.* [2012]. P^{dexp}_k contains the facts $\mathsf{r}_0(c_0), \mathsf{r}_0(c_1), \mathsf{succ}_0(c_0, c_1), \mathsf{min}_0(c_0), \mathsf{max}_0(c_1)$, and the following rules for each $i \in \{0, \dots, k-1\}$:

$$\mathsf{r}_i(x) \wedge \mathsf{r}_i(y) \rightarrow \exists z.\mathsf{s}_i(x,y,z)$$
$$\mathsf{s}_i(x,y,z) \rightarrow \mathsf{r}_{i+1}(z)$$
$$\mathsf{s}_i(x,y,z) \wedge \mathsf{s}_i(x,y',z') \wedge \mathsf{succ}_i(y,y') \rightarrow \mathsf{succ}_{i+1}(z,z')$$
$$\mathsf{s}_i(x,y,z) \wedge \mathsf{s}_i(x',y',z') \wedge \mathsf{max}_i(y) \wedge \mathsf{min}_i(y') \wedge \mathsf{succ}_i(x,x') \rightarrow \mathsf{succ}_{i+1}(z,z')$$
$$\mathsf{min}_i(x) \wedge \mathsf{s}_i(x,x,y) \rightarrow \mathsf{min}_{i+1}(y)$$
$$\mathsf{max}_i(x) \wedge \mathsf{s}_i(x,x,y) \rightarrow \mathsf{max}_{i+1}(y)$$

We extend P^{dexp}_k with the following two rules, which transitively close the precedence relation that arises from the binary predicate succ_k between the elements of the chain:

$$\mathsf{succ}_k(x,y) \rightarrow \mathsf{succt}(x,y)$$
$$\mathsf{succt}(x,y) \wedge \mathsf{succt}(y,z) \rightarrow \mathsf{succt}(x,z)$$

Next, we build $P^{\mathsf{comp}}_{\mathcal{T},w}$ which simulates the computation of $\mathcal{T}$ on w. We use the following predicates for our encoding:

- $\mathsf{state}_q(v)$ for $q \in Q$, when $\mathcal{T}$ is at state q at time v;
- $\mathsf{head}(v,x)$, when the head of $\mathcal{T}$ is on cell x at time v;
- $\mathsf{symbol}_\alpha(v,x)$ for $\alpha \in \Sigma$, when cell x contains symbol α at time v;
- $\mathsf{select}_\delta(v)$ for $\delta \in \Delta$, when the transition δ is selected at time v;
- accept, when the computation of $\mathcal{T}$ on w has reached an accepting state.

The rules of $P^{\mathsf{comp}}_{\mathcal{T},w}$ are shown in Fig. 2. Finally, let $P_{\mathcal{T},w,k} = P^{\mathsf{dexp}}_k \cup P^{\mathsf{comp}}_{\mathcal{T},w} \cup \{\mathbf{not}\ \mathsf{accept} \rightarrow \mathsf{reject}\}$, where the latter rule is added to ensure entailment of reject only when none of the computation branches reaches an accepting state. Then $P_{\mathcal{T},w,k}$ is an R-acyclic program. Indeed, the program P^{dexp}_k is clearly acyclic by construction, and its predicates do not occur in any rule head of $P^{\mathsf{comp}}_{\mathcal{T},w}$. Moreover, $P^{\mathsf{comp}}_{\mathcal{T},w}$ is R-acyclic since it does not contain existential quantifiers. The size of $P_{\mathcal{T},w,k}$ is polynomial in the size of $\mathcal{T}$, w, and k.

It is easy to see that $(*)$ holds: $\mathcal{T}$ accepts w if and only if at least one branch of the computation tree contains in its leaf an accepting state q_a. This holds if and only if at least

one of the stable models of $P_{\mathcal{T},w,k}$ contains a fact of the form $\mathsf{state}_{\mathsf{q_a}}(t)$ for some instant t; due to the acceptance rule, the latter is true if and only if there exists one stable model of $P_{\mathcal{T},w,k}$ that does not contain reject, which on its turn is true if and only if $P_{\mathcal{T},w,k} \not\models \mathsf{reject}$. □

5 Negative Reliances and R-Stratification

While positive reliances allow us to estimate if one rule can 'trigger' another rule, the use of nonmonotonic negation may also give rise to the opposite interaction where one rule 'inhibits' another. In this section, we formalise this by defining *negative reliances* between rules. This suggests a new kind of *stratification*, which generalises the classical notion but can still be decided efficiently. The impact of R-stratification on the existence of unique stable models is discussed in Section 6.

Definition 4 (Negative Reliance)**.** *Let r_1 and r_2 be rules such that $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$ and $\mathsf{sk}(r_2) = (B_2^+, B_2^-, H_2)$; w.l.o.g. assume that $\mathsf{Var}(r_1) \cap \mathsf{Var}(r_2) = \emptyset$. Rule r_2* negatively relies *on r_1 (written $r_1 \xrightarrow{-} r_2$) if there exists a set of facts F that contains no skolem terms and a substitution θ such that:*

$$B_1^+\theta \subseteq F \quad \text{(N1)} \qquad B_2^-\theta \cap H_1\theta \neq \emptyset \quad \text{(N4)}$$
$$B_1^-\theta \cap F = \emptyset \quad \text{(N2)} \qquad B_2^-\theta \cap F = \emptyset \quad \text{(N5)}$$
$$B_2^+\theta \subseteq F \quad \text{(N3)}$$

Example 3. *Consider rule $r_{(8)}$ of* (8)*, and rule $r'_{(9)}$ obtained from* (9) *by variable x with x'. We can show $r_{(8)} \xrightarrow{-} r'_{(9)}$ using $F := \{\mathsf{oH}(a), M_{\mathsf{COH}}(\mathbf{b})\} \cup \{\mathsf{hA}(a,b_i)\}_{i=1}^3$ and $\theta := \{x \mapsto a, \mathbf{y} \mapsto \mathbf{b}, x' \mapsto a\}$. Conversely, $r'_{(9)} \not\xrightarrow{-} r_{(8)}$ follows from a similar argument as in Example 1, since F is not allowed to contain skolem terms.*

The following definition is inspired by the classical notion of stratification in logic programming.

Definition 5 (R-Stratification)**.** *A sequence of disjoint programs $\mathbf{P} = P_1, \ldots, P_n$ is an* R-stratification *of a program P if $P = \bigcup_{i=1}^n P_i$ and, for every two programs $P_i, P_j \in \mathbf{P}$ and rules $r_1 \in P_i$ and $r_2 \in P_j$, we have:*

$$\text{if } r_1 \xrightarrow{\pm} r_2 \text{ then } i \leq j \qquad \text{and} \qquad \text{if } r_1 \xrightarrow{-} r_2 \text{ then } i < j.$$

P is R-stratified *if it has an R-stratification.*

Example 4. *For P consisting of rules $r_{(2)}$, $r_{(3)}$, $r_{(5)}$, $r_{(6)}$, $r_{(8)}$, and $r_{(9)}$ we obtain the reliances $r_{(2)} \xrightarrow{\pm} r_{(8)} \xrightarrow{-} r_{(9)} \xrightarrow{\pm} r_{(3)}$, $r_{(2)} \xrightarrow{\pm} r_{(3)}$, $r_{(2)} \xrightarrow{\pm} r_{(6)}$, $r_{(2)} \xrightarrow{\pm} r_{(5)} \xrightarrow{-} r_{(6)}$, $r_{(9)} \xrightarrow{\pm} r_{(5)}$, and $r_{(9)} \xrightarrow{\pm} r_{(6)}$. An R-stratification of P is therefore given by $P_1 := \{r_{(2)}, r_{(8)}\}$, $P_2 := \{r_{(3)}, r_{(5)}, r_{(9)}\}$, and $P_3 := \{r_{(6)}\}$. In contrast, P is not stratified due to rules $r_{(8)}$ and $r_{(9)}$.*

Together with the previous example, the next result shows that R-stratification properly generalises stratification.

Proposition 1. *If P is stratified, then P is R-stratified.*

Proof. Let $\mathbf{P}$ be a stratification of $P = P_1, \ldots, P_n$. We show that $\mathbf{P}$ is an R-stratification of P. Clearly, $P = \bigcup_{i=1}^n P_i$. Now consider arbitrary programs $P_i, P_j \in \mathbf{P}$ and rules $r_1 \in P_i$ and $r_2 \in P_j$. If $r_1 \xrightarrow{\pm} r_2$ then, by (P3) and (P5), there is a predicate in the head of r_1 that occurs in a positive body atom of r_2. Therefore $i \leq j$, since $\mathbf{P}$ is a stratification. Similarly, if $r_1 \xrightarrow{-} r_2$, (N4) implies that there is a predicate in the head of r_1 that occurs in a negative body atom of r_2, and thus $i < j$. □

The graph structure that is induced by reliances, defined next, can be used to decide R-stratification in practice, as shown in Proposition 2 below.

Definition 6 (Graph of Reliances)**.** *For a program P, the* graph of reliances $\mathsf{GoR}(P)$ *is a directed graph that has the rules of P as its vertices and two sets of edges:* positive edges *that correspond to the positive reliances of P and* negative edges *that correspond to the negative reliances of P.*

Proposition 2. *P is R-stratified iff its graph of reliances $\mathsf{GoR}(P)$ contains no directed cycle with a negative edge.*

Proof. Consider a program P. First, assume that $\mathsf{GoR}(P)$ has no cycles with negative edges. Let $\approx$ be an equivalence relation on P defined by setting $r_1 \approx r_2$ if r_1 and r_2 occur on a cycle of positive edges in $\mathsf{GoR}(P)$. Let $\mathsf{GoR}(P)_\approx$ be the factorisation of $\mathsf{GoR}(P)$ by $\approx$:

- vertices of $\mathsf{GoR}(P)_\approx$ are equivalence classes of $\approx$ (which we denote as $[r]_\approx := \{r' \in P \mid r \approx r'\}$);
- $\mathsf{GoR}(P)_\approx$ contains an edge $[r_1]_\approx \xrightarrow{\pm} [r_2]_\approx$ whenever $r'_1 \xrightarrow{\pm} r'_2$ for some $r_i \in [r_i]_\approx$ ($i \in \{1,2\}$);
- $\mathsf{GoR}(P)_\approx$ contains an edge $[r_1]_\approx \xrightarrow{-} [r_2]_\approx$ whenever $r'_1 \xrightarrow{-} r'_2$ for some $r_i \in [r_i]_\approx$ ($i \in \{1,2\}$).

Since any cycles in $\mathsf{GoR}(P)$ must consist of positive edges only, $\mathsf{GoR}(P)_\approx$ is a directed acyclic graph. Any topological order of the nodes of $\mathsf{GoR}(P)_\approx$ (which are sets of rules) satisfies the properties of an R-stratification.

Conversely, assume that $\mathsf{GoR}(P)$ has a cycle $r_0 \xrightarrow{\pm} \ldots \xrightarrow{\pm} r_k \xrightarrow{-} r_{k+1} \xrightarrow{\pm} \ldots \xrightarrow{\pm} r_{\ell-1} \xrightarrow{\pm} r_0$. Suppose for a contradiction that P has a stratification $P_1, \ldots, P_n$. For each rule r_i ($i = 0, \ldots, \ell-1$), let $p(i)$ denote an integer such that $r_i \in P_{p(i)}$. By the properties of R-stratification, we have $p(k) < p(k+1)$ but also $p(i) \leq p((i+1) \bmod \ell)$ for each $i \in \{0, \ldots, \ell-1\}$. The latter implies $p(k+1) \leq p(k)$—a contradiction. Hence, P has no R-stratification. □

From the previous result it is clear that, given the graph of reliances, R-stratification can be decided in polynomial time. The overall complexity is therefore dominated by the complexity of checking individual reliances—in this sense, it is polynomial in the total number of rules, and coNP-complete only in the maximal size of a rule. Moreover, in contrast to the NP-completeness of checking positive reliances (Theorem 1), negative reliances can be detected in polynomial time.

Theorem 3. *Given rules r_1 and r_2, it can be decided in polynomial time whether $r_1 \xrightarrow{-} r_2$. Checking whether a program P is R-stratified is* coNP*-complete.*

Proof. We first show that $r_1 \xrightarrow{-} r_2$ can be decided in polynomial time. Let r_1 and r_2 be two rules with $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$ and $\mathsf{sk}(r_2) = (B_2^+, B_2^-, H_2)$; w.l.o.g. we assume that $\mathsf{Var}(r_1) \cap \mathsf{Var}(r_2) = \emptyset$. To check $r_1 \xrightarrow{-} r_2$, we apply the following algorithm: for each $\alpha \in H_1$ and each $\beta \in B_2^-$, check if the following conditions are satisfied:

(i) there exists a most general unifier σ of α and β;

(ii) there are no skolem symbols in $B_1^+\sigma \cup B_2^+\sigma$;

(iii) $(B_1^-\sigma \cup B_2^-\sigma) \cap (B_1^+\sigma \cup B_2^+\sigma) = \emptyset$.

If these conditions hold for at least one pair of α and β, return $r_1 \xrightarrow{-} r_2$; else return $r_1 \not\xrightarrow{-} r_2$.

The above algorithm clearly runs in polynomial time: at most $|H_1| \times |B_2^-|$ pairs of atoms need to be considered, their most general unifier can be computed in linear time, and the remaining checks are easy to perform in polynomial time.

We now show that the algorithm is correct. For soundness, assume that the algorithm returns $r_1 \xrightarrow{-} r_2$, and let σ be the unifier considered in the algorithm when terminating. We define a substitution $\theta := \sigma \circ \theta_c$, where θ_c is the substitution that maps every variable x to a fresh constant c_x unique for x, and a set of facts $F := B_1^+\theta \cup B_2^+\theta$. It is easy to see that conditions (N1)–(N5) hold for this choice of F and θ.

To show completeness of the algorithm, assume that $r_1 \xrightarrow{-} r_2$. Then there exist F and θ satisfying the conditions of Definition 4. We can assume that $F = B_1^+\theta \cup B_2^+\theta$, which is w.l.o.g. since only conditions (N1) and (N3) require F to contain atoms. By (N4), there exists $\alpha \in H_1$ and $\beta \in B_2^-$, such that $\alpha\theta = \beta\theta$. Thus, in particular α and β have a most general unifier σ, that is, $\theta = \sigma\theta'$ for some θ'. Thus, since there are no skolem symbols in $F = B_1^+\theta \cup B_2^+\theta$ by Definition 4, condition (ii) is also satisfied. Likewise, condition (iii) follows from (N2) and (N5).

It remains to show that deciding if a program is R-stratified is coNP-complete. The proof is similar to the proof of coNP-completeness of R-acyclicity in Theorem 1. Membership is immediate by verifying the existence of a problematic cycle as in Proposition 2, which can be done in NP. Hardness was shown in Theorem 1 by constructing two rules of the form $r_1 : \mathsf{Start} \to H_1$ and $r_2 : B_2 \to \mathsf{Goal}$ that satisfy the equivalence (11). Clearly, (11) still holds if we modify r_1 to $r_1' : \mathsf{Start} \wedge \mathbf{not}\ \mathsf{Goal} \to H_1$. Then $r_2 \xrightarrow{-} r_1'$, and hence $\{r_1', r_2\}$ is R-stratified if and only if $r_1' \not\xrightarrow{+} r_2$ if and only if there is no homomorphism from Q to Q'. □

6 Computing Stable Models

In this section, we show that R-stratified programs have at most one stable model, and that this model can always be obtained by repeated application of rules according to their stratification. This leads to a semi-decision procedure for entailment. If the program is also R-acyclic, we obtain a decision procedure and tight complexity bounds.

Note that Definition 4 does not include a condition that corresponds to (P6) from Definition 2. Indeed, as the next example shows, such a condition would not lead to a notion of R-stratification that ensures unique stable models.

Example 5. *Given the rules* $r_1 : \mathbf{not}\ \mathsf{p} \to \mathsf{q}$ *and* $r_2 : \mathsf{q} \to \mathsf{p}$, *we find that* $r_1 \xrightarrow{+} r_2$ *and* $r_2 \xrightarrow{-} r_1$, *so that the program is not R-stratified. Indeed, it has no stable models for the empty set of facts. Yet, if we required that* $H_2\theta \not\subseteq F$ *in Definition 4, then* $r_2 \xrightarrow{-} r_1$ *would not hold, and the program would be R-stratified. Intuitively speaking, negative reliances do not just consider the case where* r_2 *could derive something new, but also the case where* r_2 *has already been used in a derivation that is no longer justified after applying* r_1.

We now define a computation scheme that can be used to obtain the unique stable model of R-stratified programs, or to derive a contradiction $\bot$ if no such model exists.

Definition 7. *For a set of facts* F *and a program* P *with R-stratification* $\mathbf{P} = P_1, \ldots, P_n$, *define* $S_{\mathbf{P}}^0(F) := F$ *and*

$$S_{\mathbf{P}}^{i+1}(F) := T_{P_{i+1}}^{\infty}(S_{\mathbf{P}}^i(F)) \quad \text{for} \quad 0 \le i < n.$$

For the remainder of this section, let P denote an R-stratified program with R-stratification $\mathbf{P} = P_1, \ldots, P_n$, and let F denote a set of facts. We use the abbreviations $P_1^m := \bigcup_{i=1}^m P_i$, $P_1^0 := \emptyset$, and $S_{\mathbf{P}}^i := S_{\mathbf{P}}^i(F)$.

We first show that $S_{\mathbf{P}}^n$ is a (not necessarily unique) stable model of $F \cup P$, provided that $\bot \notin S_{\mathbf{P}}^n$. The next two lemmas are key ingredients to this proof. Intuitively speaking, Lemma 1 asserts that, if the body of a rule $r \in P_i$ is satisfied at some point while computing $S_{\mathbf{P}}^i$, then it will remain satisfied in all later stages of the computation. The crucial claim is that the negative part of the rule will not be derived at any later stage. The proof of Lemma 1 relies on the definition of $\xrightarrow{-}$.

Lemma 1. *Consider numbers* $1 \le i \le j \le k \le n$ *and* $\ell \ge 0$, *a rule* $r \in P_i$ *with skolemisation* $\mathsf{sk}(r) = (B^+, B^-, H)$, *and a substitution* θ. *Then* $T_{P_j}^{\ell}(S_{\mathbf{P}}^{j-1}) \models B^+\theta, \mathbf{not}\, B^-\theta$ *implies* $S_{\mathbf{P}}^k \models B^+\theta, \mathbf{not}\, B^-\theta$.

Proof. For brevity, define $\mathcal{M} = T_{P_j}^{\ell}(S_{\mathbf{P}}^{j-1})$ and $\mathcal{M}' = S_{\mathbf{P}}^k$. Assume $\mathcal{M} \models B^+\theta, \mathbf{not}\, B^-\theta$. Suppose for a contradiction that $\mathcal{M}' \not\models B^+\theta, \mathbf{not}\, B^-\theta$. Since $\mathcal{M} \subseteq \mathcal{M}'$, we find that $\mathcal{M}' \models B^+\theta$. Hence $\mathcal{M}' \not\models \mathbf{not}\, B^-\theta$, that is, $\mathcal{M}' \cap B^-\theta \neq \emptyset$. Thus there are $m \ge j$ and $o \ge 0$ such that $T_{P_m}^o(S_{\mathbf{P}}^{m-1}) \cap B^-\theta = \emptyset$ and $T_{P_m}^{o+1}(S_{\mathbf{P}}^{m-1}) \cap B^-\theta \neq \emptyset$. Hence there is a rule $r_1 \in P_m$ with $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$, and a substitution θ_1 such that

$$B_1^+\theta_1 \subseteq T_{P_m}^o(S_{\mathbf{P}}^{m-1}) \tag{12}$$

$$B_1^-\theta_1 \cap T_{P_m}^o(S_{\mathbf{P}}^{m-1}) = \emptyset \tag{13}$$

$$H_1\theta_1 \cap B^-\theta \neq \emptyset \tag{14}$$

We show that $r_1 \xrightarrow{-} r$. For brevity, let $F' := T_{P_m}^o(S_{\mathbf{P}}^{m-1})$. If H_1 contains a skolem term $f(\mathbf{x})$, then $f(\mathbf{x})\theta_1 \notin \mathsf{Terms}(F')$. Indeed, $f(\mathbf{x})\theta_1 \in \mathsf{Terms}(F')$ would imply that $H_1\theta_1 \subseteq T_{P_m}^o(S_{\mathbf{P}}^{m-1})$, which would contradict $T_{P_m}^o(S_{\mathbf{P}}^{m-1}) \cap B^-\theta = \emptyset$.

We show that $G := \gamma_{F'}(T_{P_m}^o(S_{\mathbf{P}}^{m-1}))$ and $\sigma := \gamma_{F'}(\theta_1 \cup \theta)$ establish the conditions for $r_1 \xrightarrow{-} r$ in Definition 4:

(N1) $B_1^+\sigma \subseteq G$ by (12).

(N2) $B_1^-\sigma \cap G = \emptyset$ by (13).

(N3) $B^+\sigma \subseteq G$ by $T_{P_j}^{\ell}(S_{\mathbf{P}}^{j-1}) \models B^+\theta$ and $T_{P_j}^{\ell}(S_{\mathbf{P}}^{j-1}) \subseteq T_{P_m}^o(S_{\mathbf{P}}^{m-1})$; note that if $m = j$, then $o \ge \ell$ which follows from $T_{P_j}^{\ell}(S_{\mathbf{P}}^{j-1}) \cap B^-\theta = \emptyset$ and $T_{P_j}^{o+1}(S_{\mathbf{P}}^{j-1}) \cap B^-\theta \neq \emptyset$.

(N4) $B^-\sigma \cap H_1\sigma \neq \emptyset$ by (14); note that for every skolem term $f(\mathbf{x})\theta_1$ in $H_1\theta_1$, the definition of $\gamma_{F'}$ ensures $\gamma_{F'}(f(\mathbf{x})\theta_1) = f(\gamma_{F'}(\mathbf{x}\theta_1))$ (the former occurs in $B^-\sigma$, the latter occurs in $H_1\sigma$).

(N5) $B^-\sigma \cap G = \emptyset$ by $T_{P_m}^o(S_{\mathbf{P}}^{m-1}) \cap B^-\theta = \emptyset$.

Thus $r_1 \xrightarrow{-} r$. However, $r \in P_i$, $r_1 \in P_m$, and $i \leq j \leq m$, which contradicts Definition 5. □

Lemma 2 complements the previous result. Intuitively speaking, it states that a rule $r \in P_i$, which is clearly satisfied after computing $S^i_{\mathbf{P}}$, will remain satisfied in all later stages of the computation. The key part of this claim concerns the case that r is satisfied because its positive body is not satisfied. In this case, the positive body will never become satisfied later on, unless the head of the rule becomes satisfied as well. This argument hinges upon the definition of $\xrightarrow{+}$.

Lemma 2. *Consider numbers $1 \leq i < j \leq k \leq n$, a rule $r \in P_i$, and a substitution θ. Then $S^j_{\mathbf{P}} \models \mathsf{sk}(r)\theta$ implies $S^k_{\mathbf{P}} \models \mathsf{sk}(r)\theta$.*

Proof. Let $\mathsf{sk}(r) = (B^+, B^-, H)$ and suppose for a contradiction that $S^j_{\mathbf{P}} \models \mathsf{sk}(r)\theta$ and $S^k_{\mathbf{P}} \not\models \mathsf{sk}(r)\theta$. Since $S^j_{\mathbf{P}} \subseteq S^k_{\mathbf{P}}$, neither $B^-\theta \cap S^j_{\mathbf{P}} \neq \emptyset$ nor $S^j_{\mathbf{P}} \models (B^+ \cup H)\theta, \mathbf{not}\, B^-\theta$ may hold. Therefore, $B^+\theta \not\subseteq S^j_{\mathbf{P}}$. By $S^k_{\mathbf{P}} \not\models \mathsf{sk}(r)\theta$, we have $S^k_{\mathbf{P}} \models B^+\theta, \mathbf{not}\, B^-\theta$ and $S^k_{\mathbf{P}} \not\models H\theta$. As a consequence, there exists a maximal non-empty set of facts A such that $A \subseteq B^+\theta$ and $A \cap S^j_{\mathbf{P}} = \emptyset$. Hence there are numbers $\ell \geq 0$ and m such that $j < m \leq k$, a rule $r_1 \in P_m$ with $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$ and a substitution θ_1 such that

$$B_1^+\theta_1 \subseteq T^\ell_{P_m}(S^{m-1}_{\mathbf{P}}) \tag{15}$$

$$B_1^-\theta_1 \cap T^\ell_{P_m}(S^{m-1}_{\mathbf{P}}) = \emptyset \tag{16}$$

$$H_1\theta_1 \cap A \not\subseteq T^\ell_{P_m}(S^{m-1}_{\mathbf{P}}) \tag{17}$$

Let $F' := T^\ell_{P_m}(S^{m-1}_{\mathbf{P}}) \cup (A \setminus H_1\theta_1)$. We claim that the set of facts $G := \gamma_{F'}(T^\ell_{P_m}(S^{m-1}_{\mathbf{P}}) \cup (A \setminus H_1\theta_1))$ and substitution $\sigma := \gamma_{F'}(\theta_1 \cup \theta)$ meet the conditions for $r_1 \xrightarrow{+} r$ in Definition 2.

(P1) $B_1^+\sigma \subseteq G$ by (15).

(P2) $B_1^-\sigma \cap G = \emptyset$ by (16) and $B_1^-\theta_1 \cap A = \emptyset$; note that the latter follows from $A \subseteq S^k_{\mathbf{P}}$ and $S^k_{\mathbf{P}} \models B_1^+\theta_1, \mathbf{not}\, B_1^-\theta_1$ which is a consequence of Lemma 1, $T^\ell_{P_m}(S^{m-1}_{\mathbf{P}}) \models B_1^+\theta_1, \mathbf{not}\, B_1^-\theta_1$, $r_1 \in P_m$ and $k \geq m$.

(P3) $B^+\sigma \subseteq G \cup H_1\sigma$ by $B^+\theta \subseteq F' \cup H_1\theta_1$.

(P4) $B^-\sigma \cap (G \cup H_1\sigma) = \emptyset$ by $S^k_{\mathbf{P}} \models B^+\theta, \mathbf{not}\, B^-\theta$.

(P5) $B^+\sigma \not\subseteq G$ by (17).

(P6) $H\sigma \not\subseteq G \cup H_1\sigma$ by $G \cup H_1\theta_1 \subseteq S^k_{\mathbf{P}}$ and $S^k_{\mathbf{P}} \not\models H\theta$.

Thus, by $r_1 \in P_m$, $r \in P_i$ and $m > i$ we derive a contradiction and show our initial claim. □

Using Lemmas 1 and 2, we can show the following result.

Proposition 3. *If $\bot \notin S^n_{\mathbf{P}}$, then $S^n_{\mathbf{P}} \models_{\mathsf{SM}} F \cup P$.*

Proof. We show that $\bot \notin S^n_{\mathbf{P}}$ implies (♣) $S^n_{\mathbf{P}}$ is a model of $F \cup \mathsf{GL}(P^n_1, S^n_{\mathbf{P}})$ and (♠) the model $S^n_{\mathbf{P}}$ of $F \cup \mathsf{GL}(P^n_1, S^n_{\mathbf{P}})$ is minimal.

(♣) We prove $S^k_{\mathbf{P}} \models_{\mathsf{SM}} F \cup \mathsf{GL}(P^k_1, S^k_{\mathbf{P}})$ for all $k \in \{0, \ldots, n\}$ by induction over k.

If $k = 0$, then $P^k_1 = \emptyset$ and $S^k_{\mathbf{P}} = F$. Clearly, F is a stable model of F, since $\mathsf{GL}(\emptyset, F) = \emptyset$ and $T^\infty_\emptyset(F) = F$.

For the induction step, let $S^k_{\mathbf{P}}$ be a model of $F \cup \mathsf{GL}(P_1, S^k_{\mathbf{P}})$ (induction hypothesis). We claim that $S^{k+1}_{\mathbf{P}}$ is a model of $F \cup \mathsf{GL}(P^{k+1}_1, S^{k+1}_{\mathbf{P}})$. Note that $S^{k+1}_{\mathbf{P}} \models F \cup \mathsf{GL}(P^{k+1}_1, S^{k+1}_{\mathbf{P}})$ is equivalent to $S^{k+1}_{\mathbf{P}} \models F \cup \mathsf{sk}(P^{k+1}_1)$. Since $S^{k+1}_{\mathbf{P}} = T^\infty_{P_{k+1}}(S^k_{\mathbf{P}})$, we find $S^{k+1}_{\mathbf{P}} \models F \cup \mathsf{sk}(P_{k+1})$. It remains to show $S^{k+1}_{\mathbf{P}} \models \mathsf{sk}(P^k_1)$. Thus consider an arbitrary rule $r \in P^k_1$. By induction hypothesis, $S^k_{\mathbf{P}} \models \mathsf{sk}(P^k_1)$, and thus $S^k_{\mathbf{P}} \models \mathsf{sk}(r)$. By Lemma 2, $S^{k+1}_{\mathbf{P}} \models \mathsf{sk}(r)$. Since r was arbitrary, this shows the claim.

(♠) Suppose for a contradiction that there is $\mathcal{M} \subsetneq S^n_{\mathbf{P}}$ such that $\mathcal{M} \models F \cup \mathsf{GL}(P^n_1, S^n_{\mathbf{P}})$. Then there are $\ell \geq 0$ and j, where $1 \leq j \leq n$, such that $T^\ell_{P_j}(S^{j-1}_{\mathbf{P}}) \subseteq \mathcal{M}$ and $T^{\ell+1}_{P_j}(S^{j-1}_{\mathbf{P}}) \not\subseteq \mathcal{M}$. Thus there is a rule $r \in P_j$ with $\mathsf{sk}(r) = (B^+, B^-, H)$, and a substitution θ with

$$B^+\theta \subseteq T^\ell_{P_j}(S^{j-1}_{\mathbf{P}}) \tag{18}$$

$$B^-\theta \cap T^\ell_{P_j}(S^{j-1}_{\mathbf{P}}) = \emptyset \tag{19}$$

$$H\theta \not\subseteq \mathcal{M} \tag{20}$$

By $T^\ell_{P_j}(S^{j-1}_{\mathbf{P}}) \subseteq \mathcal{M}$ and (18), $B^+\theta \subseteq \mathcal{M}$. Together with (20) and $\mathcal{M} \models F \cup \mathsf{GL}(P^n_1, S^n_{\mathbf{P}})$, this implies $(B^+\theta, \emptyset, H\theta) \notin \mathsf{GL}(P^n_1, S^n_{\mathbf{P}})$, and thus $B^-\theta \cap S^n_{\mathbf{P}} \neq \emptyset$. However, by (18), (19), and Lemma 1, $S^n_{\mathbf{P}} \models B^+\theta, \mathbf{not}\, B^-\theta$, which yields the required contradiction. □

The next lemma captures the main statement that is needed to prove the claimed uniqueness of the stable model.

Lemma 3. *If $\mathcal{M} \models_{\mathsf{SM}} P \cup F$, then $S^n_{\mathbf{P}} = \mathcal{M}$.*

Proof. We show by induction that, for every $k \in \{0, \ldots, n\}$, we have $S^k_{\mathbf{P}} = T^\infty_{\mathsf{GL}(P^k_1, \mathcal{M})}(F)$. This establishes the claim since the latter is equal to $\mathcal{M}$ if $k = n$.

For $k = 0$, the claim is immediate: $S^0_{\mathbf{P}} = F$, $P^0_1 = \emptyset$ and $\mathsf{GL}(P^0_1, \mathcal{M}) = \emptyset$, and thus $T^\infty_{\mathsf{GL}(P^0_1, \mathcal{M})}(F) = F$.

For the induction step, assume that the claim has been shown for k. We show that $S^{k+1}_{\mathbf{P}} = T^\infty_{\mathsf{GL}(P^{k+1}_1, \mathcal{M})}(F)$.

We first show that $S^{k+1}_{\mathbf{P}} \subseteq T^\infty_{\mathsf{GL}(P^{k+1}_1, \mathcal{M})}(F)$. For brevity, we use T^i to denote $T^\infty_{\mathsf{GL}(P^i_1, \mathcal{M})}(F)$. Suppose for a contradiction that $S^{k+1}_{\mathbf{P}} \not\subseteq T^{k+1}$. Then there are numbers $m \in \{1, \ldots, k+1\}$ and $\ell \geq 0$ such that

$$T^\ell_{P_m}(S^{m-1}_{\mathbf{P}}) \subseteq T^{k+1} \tag{21}$$

$$T^{\ell+1}_{P_m}(S^{m-1}_{\mathbf{P}}) \not\subseteq T^{k+1} \tag{22}$$

By the induction hypothesis, $S^k_{\mathbf{P}} \subseteq T^{k+1}$, so (22) implies $m = k+1$. Thus there is a rule $r \in P_{k+1}$ with $\mathsf{sk}(r) = (B^+, B^-, H)$ and a substitution θ such that

$$B^+\theta \subseteq T^\ell_{P_{k+1}}(S^k_{\mathbf{P}}) \tag{23}$$

$$B^-\theta \cap T^\ell_{P_{k+1}}(S^k_{\mathbf{P}}) = \emptyset \tag{24}$$

$$H\theta \not\subseteq T^{k+1} \tag{25}$$

Together, (23), (21) (where $m = k+1$), and (25) imply

$$B^-\theta \cap T^{k+1} \neq \emptyset \tag{26}$$

Thus there is a rule r_1 with $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$, a substitution θ_1, and a number j such that

$$B_1^+\theta_1 \subseteq T^j_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F) \tag{27}$$

$$B_1^-\theta_1 \cap \mathcal{M} = \emptyset \tag{28}$$

$$B^-\theta \cap T^j_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F) = \emptyset \tag{29}$$

$$B^-\theta \cap H_1\theta_1 \neq \emptyset \tag{30}$$

Define $F' := T^j_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F) \cup T^{\ell}_{P_{k+1}}(S^k_{\mathbf{P}})$. We claim that the set of facts $G := \gamma_{F'}(F')$ and substitution $\sigma := \gamma_{F'}(\theta \cup \theta_1)$ satisfy the conditions for $r_1 \xrightarrow{-} r$ in Definition 4. By definition, G does not contain function symbols.

(N1) $B_1^+\sigma \subseteq G$ by (27).

(N2) $B_1^-\sigma \cap G = \emptyset$ by (28) and $F' \subseteq \mathcal{M}$, obtained from (21) (where $m = k+1$).

(N3) $B^+\sigma \subseteq G$ by (23).

(N4) $B^-\sigma \cap H_1\sigma \neq \emptyset$ by (30).

(N5) $B^-\sigma \cap G = \emptyset$ by (24) and (29).

Thus $r_1 \xrightarrow{-} r$, and therefore $r_1 \in P_1^k$ by Definition 5. By (28) and $T^k \subseteq \mathcal{M}$, we have $B_1^-\theta_1 \cap T^k = \emptyset$. By the induction hypothesis, $T^k = S^k_{\mathbf{P}}$, so $B_1^-\theta_1 \cap S^k_{\mathbf{P}} = \emptyset$. By (30) and (24), $H_1\theta_1 \not\subseteq S^k_{\mathbf{P}}$. Together, these observations imply that $B_1^+\theta_1 \not\subseteq S^k_{\mathbf{P}}$. Using again $T^k = S^k_{\mathbf{P}}$, we find that $B_1^+\theta_1 \not\subseteq T^k$.

From $B_1^+\theta_1 \not\subseteq T^k$ and (27), we conclude that there is a rule $r_2 \in P_{k+1}$ with $\mathsf{sk}(r_2) = (B_2^+, B_2^-, H_2)$, a substitution θ_2, and a number $o < j$ such that

$$B_2^+\theta_2 \subseteq T^o_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F) \tag{31}$$

$$B_2^-\theta_2 \cap \mathcal{M} = \emptyset \tag{32}$$

$$H_2\theta_2 \cap B_1^+\theta_1 \not\subseteq T^o_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F) \tag{33}$$

By (29) and (30), there is $\alpha \in H_1\theta_1 \setminus T^j_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F)$, and by (33), there is $\beta \in (H_2\theta_2 \cap B_1^+\theta_1) \setminus T^o_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F)$. Let $F'' := \mathcal{M} \setminus \{\alpha, \beta\}$. We claim that the set of facts $G' := \gamma_{F''}(F'')$ and substitution $\sigma' := \gamma_{F''}(\theta_1 \cup \theta_2)$ satisfy the conditions for $r_2 \xrightarrow{+} r_1$ in Definition 2. By definition, G' does not contain function symbols.

(P1) $B_2^+\sigma' \subseteq G'$ by (31) and $T^o_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F) \subseteq F''$. For the latter, note that $\beta \notin T^o_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F)$, and that $\alpha \notin T^o_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F)$ since $\alpha \notin T^j_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F)$ and $T^o_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F) \subseteq T^j_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F)$ (by $o < j$).

(P2) $B_2^-\sigma' \cap G' = \emptyset$ by (32).

(P3) $B_1^+\sigma' \subseteq G' \cup H_2\sigma'$ by (27) and $\alpha \notin T^j_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F)$.

(P4) $B_1^-\sigma' \cap (G' \cup H_2\sigma') = \emptyset$ by (28) and $H_2\theta_2 \subseteq \mathcal{M}$.

(P5) $B_1^+\sigma' \not\subseteq G'$ by our choice of β.

(P6) $H_1\sigma' \not\subseteq G' \cup H_2\sigma'$ by our choice of α and $\alpha \notin T^j_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F)$.

Thus $r_2 \xrightarrow{+} r_1$. This contradicts the assumption that $r_1 \in P_1^k$ while $r_2 \in P_{k+1}$, and thus refutes our initial assumption that $S^{k+1}_{\mathbf{P}} \not\subseteq T^{k+1}$. We have thus shown that $S^{k+1}_{\mathbf{P}} \subseteq T^{k+1}$.

For the converse, recall that $S^{k+1}_{\mathbf{P}}$ is a stable model of P_1^{k+1} (by Proposition 3 and $\bot \notin S^{k+1}_{\mathbf{P}}$ which is a consequence of $\bot \notin T^{k+1}$). Thus $S^{k+1}_{\mathbf{P}} = T^{\infty}_{\mathsf{GL}(P_1^{k+1},S^{k+1}_{\mathbf{P}})}(F)$. Now since $S^{k+1}_{\mathbf{P}} \subseteq T^{k+1} \subseteq \mathcal{M}$, we find that

$$\mathsf{GL}(P_1^{k+1}, S^{k+1}_{\mathbf{P}}) \supseteq \mathsf{GL}(P_1^{k+1}, T^{k+1}) \supseteq \mathsf{GL}(P_1^{k+1}, \mathcal{M}).$$

Thus $S^{k+1}_{\mathbf{P}} = T^{\infty}_{\mathsf{GL}(P_1^{k+1},S^{k+1}_{\mathbf{P}})}(F) \supseteq T^{\infty}_{\mathsf{GL}(P_1^{k+1},\mathcal{M})}(F) = T^{k+1}$ as required. □

Summing up, we obtain the main result of this section.

Theorem 4. *If $\bot \notin S^n_{\mathbf{P}}$, then $S^n_{\mathbf{P}}$ is the unique stable model of $F \cup P$. Otherwise $F \cup P$ does not have a stable model.*

Proof. Assume that $\bot \notin S^n_{\mathbf{P}}$. By Proposition 3, $S^n_{\mathbf{P}} \models_{\mathsf{SM}} F \cup P$. By Lemma 3, every stable model of $P \cup F$ is equal to $S^n_{\mathbf{P}}$, which is the claimed uniqueness.

If $\bot \in S^n_{\mathbf{P}}$, then let $P' := P \setminus P_C$, where P_C is the set of constraints that are in P. P' is clearly R-stratified, so $S^n_{\mathbf{P}} \setminus \bot$ is the unique stable model of $F \cup P'$. Since there exists at least one constraint $c \in P_C$ such that $S^n_{\mathbf{P}} \not\models c$, $F \cup P$ does not have a stable model. □

We can further improve the complexity results of Theorem 2 for programs that are both R-acyclic and R-stratified.

Theorem 5. *Let P be an R-acyclic R-stratified program, let F be a set of facts, and let α be a fact. Deciding $P \cup F \models \alpha$ is* 2ExpTime*-complete w.r.t. program complexity and* P*-complete w.r.t. data complexity.*

Proof. In the proof of Theorem 2, we have already shown that there is a maximal number n of terms that may occur in any stable model of an R-acyclic program P, where n is double exponential in the size of $P \cup F$, and polynomial in the size of F. If a is the maximal arity of predicates in P, and b is the number of predicate symbols, then the size of the computed stable model is bounded by $b \cdot n^a$, which is still double exponential in $P \cup F$ and polynomial in F. By Theorem 4, we can compute the unique stable model of $P \cup F$ as in Definition 7, where we only need to consider substitutions that map to the bounded set of relevant ground terms. The applicability of one ground rule over a set of facts of size at most $b \cdot n^a$ can be checked deterministically in time $b \cdot n^a$. As discussed in the proof of Theorem 2, the number of ground rules considered in each step is again double exponential in the size of $P \cup F$ and polynomial in the size of F; the number of steps has the same bound, since each rule can be applied at most once. Hence, $S^n_{\mathbf{P}}$ can be computed in time that is double exponential in the size of $P \cup F$, and polynomial in the size of F, and the claimed upper bounds for complexity follow.

Hardness for 2ExpTime w.r.t. the size of $P \cup F$ follows by the same Turing machine construction as in the proof of

Theorem 2, applied to deterministic Turing machines only. The resulting program is R-stratified since the transition rules do not contain negation for deterministic Turing machines. Hardness for P w.r.t. the size of F follows from the fact that already Datalog (i.e., rules without negation, existential quantifiers, or function symbols) is P-complete w.r.t. data complexity [Dantsin *et al.*, 2001]. □

7 Reliances under Constraints

To widen the classes of logic programs with unique stable models, it has been proposed to study stratification for a particular set of facts [Bidoit and Froidevaux, 1991]. Indeed, it might be that a program that does not have a unique stable model for all sets of facts still has a unique stable model for all sets of facts that arise in the context of a given application. On the other hand, notions that depend on a particular set of facts do not easily capture a wider class of relevant sets of facts, making it hard to develop logic programs that are robust to changing inputs.

In this section, we therefore propose a generalisation of R-acyclicity and R-stratification that considers *constraints*, that is, rules of the form $B^+ \rightarrow \bot$ where B^+ is a set of atoms. As illustrated by the following example, constraints restrict the possible types of input so that more programs are stratified.

Example 6. *Organic molecules are those containing carbon and each inorganic entity is a molecule of geological origin:*

$$\begin{aligned} r_1:&\quad \mathsf{mol}(x) \wedge \mathsf{hA}(x,y) \wedge \mathsf{c}(y) \rightarrow \mathsf{organic}(x)\\ r_2:&\quad \mathsf{mol}(x) \wedge \mathbf{not}\ \mathsf{organic}(x) \rightarrow \mathsf{inorganic}(x)\\ r_3:&\quad \mathsf{inorganic}(x) \rightarrow \mathsf{mol}(x) \wedge \mathsf{geoOrigin}(x) \end{aligned}$$

It is easily checked that $r_1 \dashrightarrow r_2 \xrightarrow{\pm} r_3 \xrightarrow{\pm} r_1$, so $\{r_1, r_2, r_3\}$ is not R-stratified by Proposition 2. Although the program has a unique stable model for all sets of facts, there is no stratified order of rule applications that produces the stable model. In particular, the set of facts $\{\mathsf{inorganic}(a), \mathsf{hA}(a,b), \mathsf{c}(b)\}$ requires us to apply r_3 before r_1. This situation is undesired, since inorganic molecules usually do not contain carbon, and a refined notion of reliance should take this into account.

Definition 8 (Reliances under Constraints)**.** *Let r_1 and r_2 be rules, and let C be a set of constraints.*

- *r_2* positively relies on *r_1* under *C (written $r_1 \xrightarrow{\pm}_C r_2$) if there exists a set of facts F and a substitution θ that satisfy the conditions in Definition 2, and where $F \models C$.*
- *r_2* negatively relies on *r_1* under *C (written $r_1 \dashrightarrow_C r_2$) if there exists a set of facts F and a substitution θ that satisfy the conditions in Definition 4, and where $F \models C$.*

The classes of programs that are R-acyclic under *C and* R-stratified under *C are defined as in Definition 3 and 5, respectively, but using $\xrightarrow{\pm}_C$ instead of $\xrightarrow{\pm}$.*

It should be noted that our earlier results treat constraints like any other rule of P. This is still possible here, e.g., if some constraints are not deemed to be relevant for showing stratification. Indeed, the fewer constraints are part of C, the fewer additional checks are needed to compute reliances.

Example 7. *Consider the rules of Example 6 and the constraint $c : \mathsf{inorganic}(x) \wedge \mathsf{hA}(x,y) \wedge \mathsf{c}(y) \rightarrow \bot$. With $C := \{c\}$, we find $r_3 \not\xrightarrow{\pm}_C r_1$, and indeed $P_1 := \{r_1\}$, $P_2 := \{r_2, r_3\}$ is an R-stratification under these constraints.*

The consideration of constraints increases the complexity of checking positive reliances from NP to Σ_2^P, i.e., the check can be performed in polynomial time by a nondeterministic Turing machine using an NP oracle. Yet, as before, the NP computations correspond to checking the applicability of a rule or constraint to a small set of facts, for which efficient implementations exist. A lower bound can be shown by reducing satisfiability of a quantified Boolean formula $\exists\mathbf{p}.\forall\mathbf{q}.\varphi$ to testing a positive reliance under a set of constraints.

Theorem 6. *Given rules r_1 and r_2, and a set of constraints C, deciding whether $r_1 \xrightarrow{\pm}_C r_2$ is Σ_2^P-complete. Checking whether a program P is R-acyclic under constraints is Π_2^P-complete.*

Proof. We first show the complexity for deciding $r_1 \xrightarrow{\pm}_C r_2$.

Membership: As in the proof of Theorem 1, we can guess a set of facts F and a substitution θ with size polynomial in the input and then check in polynomial time whether the conditions of Definition 2 are met. Given a constraint $c \in C$, we can check $F \models c$ by invoking a (co)NP oracle that decides the existence of a substitution under which the body of c is included in F (which would show $F \not\models c$). Since the number of oracle calls is linear in $|C|$, the overall check is in Σ_2^P.

Hardness: We consider quantified Boolean formulae (QBF) of the form $\exists\mathbf{p}.\forall\mathbf{q}.\varphi$, where $\mathbf{p}$ and $\mathbf{q}$ are lists of propositional variables, and φ is a propositional formula over $\mathbf{p} \cup \mathbf{q}$. Deciding the satisfiability of such a QBF is Σ_2^P-hard, even if it is of the form

$$\exists\mathbf{p}.\forall\mathbf{q}.(\ell_{11} \wedge \ell_{12} \wedge \ell_{13}) \vee \ldots \vee (\ell_{n1} \wedge \ell_{n2} \wedge \ell_{n3})$$

where each ℓ_{ij} is a propositional variable or a negated propositional variable [Bauer *et al.*, 1973]. Given a QBF $\exists\mathbf{p}.\forall\mathbf{q}.\varphi$ of this form, we construct rules r_1 and r_2, and constraints C such that

$$\exists\mathbf{p}.\forall\mathbf{q}.\varphi \text{ is satisfiable} \quad \Leftrightarrow \quad r_1 \xrightarrow{\pm}_C r_2. \tag{34}$$

Let $\mathbf{p} = \langle p_1, \ldots, p_k \rangle$ and $\mathbf{q} = \langle q_1, \ldots, q_m \rangle$. Rules r_1 and r_2 are defined as follows, where 0 and 1 are constants, expressions and all other terms are variables:

$$\begin{aligned} r_1:&\quad \mathsf{V}(x_1,\ldots,x_k) \wedge\\ &\quad \mathsf{N}(0,0,0) \wedge \mathsf{N}(0,0,1) \wedge \mathsf{N}(0,1,0) \wedge\\ &\quad \mathsf{N}(0,1,1) \wedge \mathsf{N}(1,0,0) \wedge \mathsf{N}(1,0,1) \wedge \mathsf{N}(1,1,0) \wedge\\ &\quad \mathsf{L}_{\mathsf{pos}}(0,0) \wedge \mathsf{L}_{\mathsf{pos}}(1,1) \wedge \mathsf{L}_{\mathsf{neg}}(0,1) \wedge \mathsf{L}_{\mathsf{neg}}(1,0) \quad \rightarrow\\ &\quad \exists v.\mathsf{B}_1(v,1,x_1) \wedge \mathsf{B}_1(v,0,x_1) \wedge \ldots \wedge \mathsf{B}_k(v,1,x_k) \wedge \mathsf{B}_k(v,0,x_k)\\ r_2:&\quad \mathsf{B}_1(w,y_1,y_1) \wedge \ldots \wedge \mathsf{B}_k(w,y_k,y_k) \rightarrow \mathsf{Goal} \end{aligned}$$

Facts matching the atom $\mathsf{V}(x_1,\ldots,x_k)$ in r_1 will be interpreted as truth assignments for $\mathbf{p}$. Facts about N enumerate all cases in which a conjunction of three truth values evaluates to false, while facts about $\mathsf{L}_{\mathsf{pos}}$ and $\mathsf{L}_{\mathsf{neg}}$ yield a convenient way to compute the truth value of positive and negative propositional literals. The predicates B_i are used to constrain the possible truth assignments: the body of r_2 can only match the head of r_1 if each variable x_i is mapped to 0 or 1.

To define C, we consider variables $z_1,\ldots,z_k$ (representing $\mathbf{p}$), $z'_1,\ldots,z'_m$ (representing $\mathbf{q}$), and $u_{11},u_{12},u_{13},\ldots,u_{n1},u_{n2},u_{n3}$ (representing truth values of literals). Given any literal ℓ_{ij} of the QBF, let $\mathsf{L}_{ij} := \mathsf{L}_{\mathsf{pos}}$ if ℓ_{ij} is a positive literal and $\mathsf{L}_{ij} := \mathsf{L}_{\mathsf{neg}}$ otherwise, and let $\xi_{ij} := z_o$ if ℓ_{ij} contains the existentially quantified propositional variable p_o and $\xi_{ij} := z'_o$ if ℓ_{ij} contains the universally quantified propositional variable q_o. We define C to consist of a single constraint c:

$$\begin{aligned} c\colon \quad & \mathsf{V}(z_1,\ldots,z_k) \wedge \\ & \mathsf{N}(u_{11},u_{12},u_{13}) \wedge \ldots \wedge \mathsf{N}(u_{n1},u_{n2},u_{n3}) \wedge \\ & \mathsf{L}_{11}(\xi_{11},u_{11}) \wedge \mathsf{L}_{12}(\xi_{12},u_{12}) \wedge \mathsf{L}_{13}(\xi_{13},u_{13}) \wedge \\ & \ldots \\ & \mathsf{L}_{n1}(\xi_{n1},u_{n1}) \wedge \mathsf{L}_{n2}(\xi_{n2},u_{n2}) \wedge \mathsf{L}_{n3}(\xi_{n3},u_{n3}) \quad \rightarrow \bot \end{aligned}$$

First, we show $\Rightarrow$ of (34). Assume that $\exists\mathbf{p}.\forall\mathbf{q}.\varphi$ is satisfiable, i.e., there exists a truth assignment $\nu_\exists : \mathbf{p} \rightarrow \{0,1\}$ such that for every truth assignment $\nu_\forall : \mathbf{q} \rightarrow \{0,1\}$, we have $\nu_\exists \cup \nu_\forall \models \varphi$. We define a substitution θ by setting $\theta(x_i) := \nu_\exists(p_i)$ and a set of facts F as follows:

$$\begin{aligned} F := & \{\mathsf{V}(\nu_\exists(p_1),\ldots,\nu_\exists(p_k))\} \cup \\ & \{\mathsf{L}_{\mathsf{pos}}(0,0),\mathsf{L}_{\mathsf{pos}}(1,1),\mathsf{L}_{\mathsf{neg}}(0,1),\mathsf{L}_{\mathsf{neg}}(1,0)\} \cup \\ & \{\mathsf{N}(\nu_1,\nu_2,\nu_3) \mid \nu_1,\nu_2,\nu_3 \in \{0,1\}\} \setminus \{\mathsf{N}(1,1,1)\} \end{aligned}$$

It is easy to check that conditions (P1)–(P6) are satisfied. Suppose for a contradiction that $F \not\models c$, and let ψ denote the body of c. Then there exists a substitution σ for the variables in c, such that $\psi\sigma \subseteq F$. Since $V(z_1,\ldots,z_k) \in \psi$, we obtain $\nu_\exists(p_i) = \sigma(z_i)$ for all $i = 1,\ldots,k$. Similarly, define $\nu_\forall : \mathbf{q} \rightarrow \{0,1\}$ by setting $\nu_\forall(q_i) := \sigma(z'_i)$. Clearly, the truth value for every literal ℓ_{ij} under $\nu_\exists \cup \nu_\forall$ is $\sigma(u_{ij})$. By $\psi\sigma \subseteq F$, we find $\nu_\exists \cup \nu_\forall \not\models \ell_{i1} \wedge \ell_{i2} \wedge \ell_{i3}$ for all $i-1,\ldots,n$. Thus $\nu_\exists \cup \nu_\forall \not\models \varphi$, which contradicts the assumption on $\nu_\exists$. Thus $F \models c$ as required.

Next, we show $\Leftarrow$ of (34). Assume that $r_1 \stackrel{+}{\rightarrow}_C r_2$ is witnessed by a set of facts F and substitution θ. By (P1), $\mathsf{V}(x_1,\ldots,x_k)\theta \in F$, and, by (P3) and (P5), we have $\mathsf{B}_i(w,y_i,y_i)\theta \in \{\mathsf{B}_i(f(\mathbf{x}),1,x_i)\theta,\mathsf{B}_i(f(\mathbf{x}),0,x_i)\theta\}$ for every $i \in \{1,\ldots,k\}$, where f is the skolem function used when skolemising r_1. This implies that $\theta(x_i) \in \{0,1\}$ for all $i \in \{1,\ldots,k\}$. We define $\nu_\exists : \mathbf{p} \rightarrow \{0,1\}$ by setting $\nu_\exists(p_i) := \theta(x_i)$. Suppose for a contradiction that there is a truth assignment $\nu_\forall : \mathbf{q} \rightarrow \{0,1\}$ such that $\nu_\exists \cup \nu_\forall \not\models \varphi$. Let σ be the substitution for the variables in c defined by setting $\sigma(z_i) := \nu_\exists(p_i)$, $\sigma(z'_i) := \nu_\forall(q_i)$, and $\sigma(u_{ij}) = 1$ if $\nu_\exists \cup \nu_\forall \models \ell_{ij}$ and $\sigma(u_{ij}) = 0$ otherwise. It is easy to see that $\psi\sigma \subseteq F$, where ψ is the body of c as above. This contradicts $r_1 \stackrel{+}{\rightarrow}_C r_2$, so we conclude that no such $\nu_\forall$ exists, i.e., that $\exists\mathbf{p}.\forall\mathbf{q}.\varphi$ is not satisfiable.

Finally, we show that checking R-acyclicity under constraints is Π^P_2-complete. Membership follows since it can be checked in Σ^P_2 that P is *not* R-acyclic under constraints, analogously to the case of R-acyclicity in Theorem 1. For hardness, consider the rules r_1 and r_2, and constraints C as constructed in the above hardness proof, as well as the rule $r_3 : \mathsf{Goal} \rightarrow \mathsf{Start}$. Let r'_1 be obtained from r_1 by adding Start to its body atoms. Clearly, the program $\{r'_1,r_2,r_3\}$ is R-acyclic under C if and only if $\exists\mathbf{p}.\forall\mathbf{q}.\varphi$ is satisfiable. □

As before, the relations $\stackrel{+}{\rightarrow}_C$ and $\stackrel{=}{\rightarrow}_C$ induce a graph of reliances under constraints. Analogously to Proposition 2, we can show that P is R-stratified under constraints if and only if this graph does not contain cycles that involve $\stackrel{=}{\rightarrow}_C$. This is the basis for deciding R-stratification under constraints, leading to the following result.

Theorem 7. *Given rules r_1 and r_2, and a set of constraints C, the problem of deciding whether $r_1 \stackrel{=}{\rightarrow}_C r_2$ is in Δ^P_2. Checking whether a program P is R-stratified under C is Π^P_2-complete.*

Proof. In the proof of Theorem 3, we showed that $r_1 \stackrel{=}{\rightarrow} r_2$ can be decided in polynomial time by describing a polynomial, correct and complete algorithm. We extend the same algorith to check $r_1 \stackrel{=}{\rightarrow}_C r_2$, by checking for each F and θ such that $r_1 \stackrel{=}{\rightarrow} r_2$ (F and θ as defined in the correctness part of Theorem 3 proof) whether $F \models C$. In order to decide whether $F \models C$, we need to check for each $c \in C$ whether $F \models c$, which requires a call to an NP-oracle. Since the number of possible F and θ pairs that the algorithm tests is polynomial in the size of the input, the algorithm needs at most a polynomial number of steps, each of which invokes a polynomial number of calls to an NP-oracle. As a consequence, a polynomial number of calls to an NP-oracle is required overall and the problem lies in Δ^P_2.

It remains to show that deciding R-stratification of a program P under a set of constraints C is Π^P_2-complete. Membership follows by the fact that it can be checked in Σ^P_2whether a program P is *not* R-stratified under a set of constraints C. Indeed, as discussed above, this can be detected by finding a cycle of the form $r_0 \stackrel{+}{\rightarrow}_C \ldots \stackrel{+}{\rightarrow}_C r_k \stackrel{=}{\rightarrow}_C r_{k+1} \stackrel{+}{\rightarrow}_C \ldots \stackrel{+}{\rightarrow}_C r_{\ell-1} \stackrel{+}{\rightarrow}_C r_0$, where $\ell \leq |P|$. One can guess such a cycle and justifications F_i, θ_i for each of the reliances $r_i \stackrel{+}{\rightarrow}_C r_{(i+1) \bmod \ell}$, and these choices can be verified by a polynomial number of calls to an NP-oracle.

For hardness, we show that checking satisfiability of a quantified Boolean formula $\exists\mathbf{p}.\forall\mathbf{q}.\varphi$ can be reduced to checking whether a program P is *not* R-stratified under a set of constraints C. In the proof of Theorem 6, we constructed a set of constraints C and rules $r_1 : B_1 \rightarrow H_1$ and $r_2 : B_2 \rightarrow \mathsf{Goal}$ satisfying (34). If we modify r_1 to $r'_1 : B_1 \wedge \mathbf{not}\ \mathsf{Goal} \rightarrow H_1$, then $\exists\mathbf{p}.\forall\mathbf{q}.\varphi$ is satisfiable if and only if $\{r'_1,r_2\}$ is not R-stratified under C, which proves our claim. □

Given an R-stratification of P under constraints C, we can again define a computation scheme to obtain unique stable models. C in this case is evaluated on all strata, though one can also defer constraint checking to the highest stratum.

Definition 9. *For a set of facts F and a program P with R-stratification $\mathbf{P} = P_1,\ldots,P_n$ under constraints C, define $S^0_{\mathbf{P},C}(F) := T_C(F)$ and*

$$S^{i+1}_{\mathbf{P},C}(F) := T^\infty_{P_{i+1} \cup C}(S^i_{\mathbf{P},C}(F)) \quad \textit{for} \quad 0 \leq i < n.$$

In the rest of the section, we abbreviate $S^i_{\mathbf{P},C}(F)$ with $S^i_{\mathbf{P},C}$.

The following result can be shown using the same overall proof structure as in Section 6. The main difference is that in all arguments that discuss potential reliances between rules,

we also need to show satisfaction of the constraints. This is usually a consequence of the assumption that $\bot$ is not derived.

Theorem 8. *If $\bot \notin S^n_{\mathbf{P},C}(F)$, then $S^n_{\mathbf{P},C}(F)$ is the unique stable model of $F \cup P \cup C$, or else $F \cup P \cup C$ has no stable model.*

We first show some intermediate results that we will use for the proof of the main claim.

Lemma 4. *Consider numbers $1 \le i \le j \le k \le n$, a rule $r \in P_i$ with $\mathsf{sk}(r) = (B^+, B^-, H)$, a substitution θ, and some $\ell \ge 0$, such that $\bot \notin S^k_{\mathbf{P},C}$. Then $T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \models B^+\theta, \mathbf{not}\, B^-\theta$ implies $S^k_{\mathbf{P},C} \models B^+\theta, \mathbf{not}\, B^-\theta$.*

Proof. For brevity, define $\mathcal{M} = T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C})$ and $\mathcal{M}' = S^k_{\mathbf{P},C}$. Assume $\mathcal{M} \models B^+\theta, \mathbf{not}\, B^-\theta$. Suppose for a contradiction that $\mathcal{M}' \not\models B^+\theta, \mathbf{not}\, B^-\theta$. Since $\mathcal{M} \subseteq \mathcal{M}'$, we find that $\mathcal{M}' \models B^+\theta$. Hence $\mathcal{M}' \not\models \mathbf{not}\, B^-\theta$, that is, $\mathcal{M}' \cap B^-\theta \neq \emptyset$. Thus there are $m \ge j$ and $o \ge 0$ such that $T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \cap B^-\theta = \emptyset$ and $T^{o+1}_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \cap B^-\theta \neq \emptyset$. Hence there is a rule $r_1 \in P_m$ with $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$, and a substitution θ_1 such that

$$B_1^+\theta_1 \subseteq T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \tag{35}$$

$$B_1^-\theta_1 \cap T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) = \emptyset \tag{36}$$

$$H_1\theta_1 \cap B^-\theta \neq \emptyset \tag{37}$$

We show that $r_1 \xrightarrow{-}_C r$. For brevity, let $F' := T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C})$. If H_1 contains a skolem term $f(\mathbf{x})$, then $f(\mathbf{x})\theta_1 \notin \mathsf{Terms}(F')$. Indeed, having $f(\mathbf{x})\theta_1 \in \mathsf{Terms}(F')$ would imply that $H_1\theta_1 \subseteq T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C})$, which would subsequently contradict $T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \cap B^-\theta = \emptyset$.

We show that $G := \gamma_{F'}(T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}))$ and $\sigma := \gamma_{F'}(\theta_1 \cup \theta)$ establish the conditions for $r_1 \xrightarrow{-}_C r$ in Definition 4:

(N1) $B_1^+\sigma \subseteq G$ by (35).

(N2) $B_1^-\sigma \cap G = \emptyset$ by (36).

(N3) $B^+\sigma \subseteq G$ is a consequence of $T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \models B^+\theta$ and $T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \subseteq T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C})$; note that if $m = j$, then $o \ge \ell$ which follows from $T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \cap B^-\theta = \emptyset$ and $T^{o+1}_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \cap B^-\theta \neq \emptyset$.

(N4) $B^-\sigma \cap H_1\sigma \neq \emptyset$ by (37); note that for every skolem term $f(\mathbf{x})\theta_1$ in $H_1\theta_1$, the definition of $\gamma_{F'}$ ensures $\gamma_{F'}(f(\mathbf{x})\theta_1) = f(\gamma_{F'}(\mathbf{x}\theta_1))$ (the former occurs in $B^-\sigma'$, the latter occurs in $H_1\sigma'$).

(N5) $B^-\sigma \cap G = \emptyset$ by $T^o_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \cap B^-\theta = \emptyset$.

Since $\bot \notin G$, we have $G \models C$. Thus $r_1 \xrightarrow{-}_C r$. However, $r \in P_i$, $r_1 \in P_m$, and $i \le j \le m$, which contradict Definition 5. □

Lemma 5. *Consider numbers $1 \le i < j \le k \le n$, a rule $r \in P_i$, and a substitution θ, such that $\bot \notin S^k_{\mathbf{P},C}$. Then $S^j_{\mathbf{P},C} \models \mathsf{sk}(r)\theta$ implies $S^k_{\mathbf{P},C} \models \mathsf{sk}(r)\theta$.*

Proof. Let $\mathsf{sk}(r) = (B^+, B^-, H)$ and suppose for a contradiction that $S^j_{\mathbf{P},C} \models \mathsf{sk}(r)\theta$ and $S^k_{\mathbf{P},C} \not\models \mathsf{sk}(r)\theta$. Since $S^j_{\mathbf{P},C} \subseteq S^k_{\mathbf{P},C}$, we cannot have either $B^-\theta \cap S^j_{\mathbf{P},C} \neq \emptyset$ or $S^j_{\mathbf{P},C} \models (B^+ \cup H)\theta, \mathbf{not}\, B^-\theta$. So, $B^+\theta \not\subseteq S^j_{\mathbf{P},C}$. By $S^k_{\mathbf{P},C} \not\models \mathsf{sk}(r)\theta$, we derive $S^k_{\mathbf{P},C} \models B^+\theta, \mathbf{not}\, B^-\theta$ and $S^k_{\mathbf{P},C} \not\models H\theta$. Therefore, there exists non-empty set of facts A such that $A \subseteq B^+\theta$ and $A \cap S^j_{\mathbf{P},C} = \emptyset$. Hence there are numbers $\ell \ge 0$ and m such that $j < m \le k$, a rule $r_1 \in P_m$ with $\mathsf{sk}(r_1) = (B_1^+, B_1^-, H_1)$ and a substitution θ_1 such that

$$B_1^+\theta_1 \subseteq T^\ell_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \tag{38}$$

$$B_1^-\theta_1 \cap T^\ell_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) = \emptyset \tag{39}$$

$$H_1\theta_1 \cap A \not\subseteq T^\ell_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \tag{40}$$

Let $F' := T^\ell_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \cup (A \setminus H_1\theta_1)$. We claim that the set of facts $G := \gamma_{F'}(T^\ell_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \cup (A \setminus H_1\theta_1))$ and substitution $\sigma := \gamma_{F'}(\theta_1 \cup \theta)$ meet the conditions for $r_1 \xrightarrow{+}_C r$ in Definition 2.

(P1) $B_1^+\sigma \subseteq G$ by (38).

(P2) $B_1^-\sigma \cap G = \emptyset$ by (39) and $B_1^-\theta_1 \cap A = \emptyset$; note that the latter follows from $A \subseteq S^k_{\mathbf{P},C}$ and $S^k_{\mathbf{P},C} \models B_1^+\theta_1, \mathbf{not}\, B_1^-\theta_1$ which is a consequence of Lemma 4, $T^\ell_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \models B_1^+\theta_1, \mathbf{not}\, B_1^-\theta_1$, $r_1 \in P_m$ and $k \ge m$.

(P3) $B^+\sigma \subseteq G \cup H_1\sigma$ by $B^+\theta \subseteq F' \cup H_1\theta_1$.

(P4) $B^-\sigma \cap (G \cup H_1\sigma) = \emptyset$ by $S^k_{\mathbf{P},C} \models B^+\theta, \mathbf{not}\, B^-\theta$.

(P5) $B^+\sigma \not\subseteq G$ by (40).

(P6) $H\sigma \not\subseteq G \cup H_1\sigma$ by $G \cup H_1\theta_1 \subseteq S^k_{\mathbf{P},C}$ and $S^k_{\mathbf{P},C} \not\models H\theta$.

Also $\bot \notin G$, so, $G \models C$. Thus, by $r_1 \in P_m$, $r \in P_i$ and $m > i$ we derive a contradiction and show our initial claim. □

Proposition 4. *If $\bot \notin S^n_{\mathbf{P},C}$, then $S^n_{\mathbf{P},C} \models_{\mathsf{SM}} F \cup P \cup C$.*

Proof. We show that $\bot \notin S^n_{\mathbf{P},C}$ implies (♣) $S^n_{\mathbf{P},C}$ is a model of $F \cup \mathsf{GL}(P^n_1 \cup C, S^n_{\mathbf{P},C})$ and (♠) the model $S^n_{\mathbf{P},C}$ of $F \cup \mathsf{GL}(P^n_1 \cup C, S^n_{\mathbf{P},C})$ is minimal.

(♣) We prove $S^k_{\mathbf{P},C} \models_{\mathsf{SM}} F \cup \mathsf{GL}(P^k_1 \cup C, S^k_{\mathbf{P},C})$ for all $k \in \{0, \ldots, n\}$ by induction over k.

If $k = 0$, then $P^k_1 = \emptyset$ and $S^k_{\mathbf{P},C} = F$ (since $\bot \notin S^n_{\mathbf{P},C}$). Clearly, F is a stable model of $F \cup \mathsf{GL}(C, F)$, since $T^\infty_{\mathsf{GL}(C,F)}(F) = F$.

For the induction step, let $S^k_{\mathbf{P},C}$ be a model of $F \cup \mathsf{GL}(P^k_1 \cup C, S^k_{\mathbf{P},C})$ (induction hypothesis). We claim that $S^{k+1}_{\mathbf{P},C}$ is a model of $F \cup \mathsf{GL}(P^{k+1}_1 \cup C, S^{k+1}_{\mathbf{P},C})$. Note that $S^{k+1}_{\mathbf{P},C} \models F \cup \mathsf{GL}(P^{k+1}_1 \cup C, S^{k+1}_{\mathbf{P},C})$ is equivalent to $S^{k+1}_{\mathbf{P},C} \models F \cup \mathsf{sk}(P^{k+1}_1 \cup C)$. Since $S^{k+1}_{\mathbf{P},C} = T^\infty_{P_{k+1} \cup C}(S^k_{\mathbf{P},C})$, we find $S^{k+1}_{\mathbf{P},C} \models F \cup \mathsf{sk}(P_{k+1})$. It remains to show $S^{k+1}_{\mathbf{P},C} \models \mathsf{sk}(P^k_1 \cup C)$. Thus consider an arbitrary rule $r \in P^k_1$. By induction hypothesis, $S^k_{\mathbf{P},C} \models \mathsf{sk}(P^k_1 \cup C)$, and thus

$S^k_{\mathbf{P},C} \models \mathsf{sk}(r)$. By Lemma 5, $S^{k+1}_{\mathbf{P},C} \models \mathsf{sk}(r)$. Since r was arbitrary, this shows the claim.

($\spadesuit$) Suppose for a contradiction that there is $\mathcal{M} \subsetneq S^n_{\mathbf{P},C}$ such that $\mathcal{M} \models F \cup \mathsf{GL}(P^n_1 \cup C, S^n_{\mathbf{P},C})$. Then there are $\ell \geq 0$ and j, where $1 \leq j \leq n$, such that $T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \subseteq \mathcal{M}$ and $T^{\ell+1}_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \not\subseteq \mathcal{M}$. Thus there is a rule $r \in P_j$ with $\mathsf{sk}(r) = (B^+, B^-, H)$, and a substitution θ with

$$B^+\theta \subseteq T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \tag{41}$$

$$B^-\theta \cap T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) = \emptyset \tag{42}$$

$$H\theta \not\subseteq \mathcal{M} \tag{43}$$

By $T^\ell_{P_j \cup C}(S^{j-1}_{\mathbf{P},C}) \subseteq \mathcal{M}$ and (41), $B^+\theta \subseteq \mathcal{M}$. Together with (43) and $\mathcal{M} \models F \cup \mathsf{GL}(P^n_1 \cup C, S^n_{\mathbf{P},C})$, this implies $(B^+\theta, \emptyset, H\theta) \notin \mathsf{GL}(P^n_1 \cup C, S^n_{\mathbf{P},C})$, and thus $B^-\theta \cap S^n_{\mathbf{P},C} \neq \emptyset$. However, by (41), (42), and Lemma 4, $S^n_{\mathbf{P},C} \models B^+\theta, \mathbf{not}\, B^-\theta$, which yields the required contradiction. □

Lemma 6. *If $\mathcal{M} \models_{\mathsf{SM}} F \cup P \cup C$, then $S^n_{\mathbf{P},C} = \mathcal{M}$.*

Proof. We show by induction that, for every $k \in \{0, \ldots, n\}$, we have $S^k_{\mathbf{P},C} = T^\infty_{\mathsf{GL}(P^k_1 \cup C, \mathcal{M})}(F)$. This establishes the claim since the latter is equal to $\mathcal{M}$ if $k = n$.

For $k = 0$, we have $S^0_{\mathbf{P},C} = F$ and $T^\infty_{\mathsf{GL}(P^0_1 \cup C, \mathcal{M})}(F) = C(F)$. Since $\mathcal{M} \models_{\mathsf{SM}} F \cup P \cup C$, $\bot \notin C(F)$.

For the induction step, assume that the claim has been shown for k. We show that $S^{k+1}_{\mathbf{P},C} = T^\infty_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$.

We first show that $S^{k+1}_{\mathbf{P},C} \subseteq T^\infty_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$. For brevity, we use T^i to denote $T^\infty_{\mathsf{GL}(P^i_1 \cup C, \mathcal{M})}(F)$. Suppose for a contradiction that $S^{k+1}_{\mathbf{P},C} \not\subseteq T^{k+1}$. Then there are numbers $m \in \{1, \ldots, k+1\}$ and $\ell \geq 0$ such that

$$T^\ell_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \subseteq T^{k+1} \tag{44}$$

$$T^{\ell+1}_{P_m \cup C}(S^{m-1}_{\mathbf{P},C}) \not\subseteq T^{k+1} \tag{45}$$

By the induction hypothesis, $S^k_{\mathbf{P},C} \subseteq T^{k+1}$, so (45) implies $m = k+1$. Thus there is a rule $r \in P_{k+1}$ with $\mathsf{sk}(r) = (B^+, B^-, H)$ and a substitution θ such that

$$B^+\theta \subseteq T^\ell_{P_{k+1} \cup C}(S^k_{\mathbf{P},C}) \tag{46}$$

$$B^-\theta \cap T^\ell_{P_{k+1} \cup C}(S^k_{\mathbf{P},C}) = \emptyset \tag{47}$$

$$H\theta \not\subseteq T^{k+1} \tag{48}$$

Together, (46), (44) (where $m = k+1$), and (48) imply

$$B^-\theta \cap T^{k+1} \neq \emptyset \tag{49}$$

Thus there is a rule r_1 with $\mathsf{sk}(r_1) = (B^+_1, B^-_1, H_1)$, a substitution θ_1, and a number j such that

$$B^+_1\theta_1 \subseteq T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F) \tag{50}$$

$$B^-_1\theta_1 \cap \mathcal{M} = \emptyset \tag{51}$$

$$B^-\theta \cap T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F) = \emptyset \tag{52}$$

$$B^-\theta \cap H_1\theta_1 \neq \emptyset \tag{53}$$

Define $F' := T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F) \cup T^\ell_{P_{k+1} \cup C}(S^k_{\mathbf{P},C})$. We claim that the set of facts $G := \gamma_{F'}(F')$ and substitution $\sigma := \gamma_{F'}(\theta \cup \theta_1)$ satisfy the conditions for $r_1 \xrightarrow{-}_C r$ in Definition 4. By definition, G does not contain function symbols.

(N1) $B^+_1\sigma \subseteq G$ by (50).

(N2) $B^-_1\sigma \cap G = \emptyset$ by (51) and $F' \subseteq \mathcal{M}$, obtained from (44) (where $m = k+1$).

(N3) $B^+\sigma \subseteq G$ by (46).

(N4) $B^-\sigma \cap H_1\sigma \neq \emptyset$ by (53).

(N5) $B^-\sigma \cap G = \emptyset$ by (47) and (52).

By $\mathcal{M} \models F \cup P \cup C$ and $T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F) \subseteq \mathcal{M}$, we have $\bot \notin T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$; also, by (44) (where $m = k+1$) $T^\ell_{P_{k+1} \cup C}(S^k_{\mathbf{P},C}) \subseteq T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$ and, so, $\bot \notin T^\ell_{P_{k+1} \cup C}(S^k_{\mathbf{P},C})$. Therefore, $\bot \notin F'$ and, thus, $\bot \notin G$, which implies $G \models C$. As a consequence, $r_1 \xrightarrow{-} r$, and therefore $r_1 \in P^k_1$ by Definition 5. By (51) and $T^k \subseteq \mathcal{M}$, we have $B^-_1\theta_1 \cap T^k = \emptyset$. By the induction hypothesis, $T^k = S^k_{\mathbf{P},C}$, so $B^-_1\theta_1 \cap S^k_{\mathbf{P},C} = \emptyset$. By (53) and (47), $H_1\theta_1 \not\subseteq S^k_{\mathbf{P},C}$. Together, these observations imply that $B^+_1\theta_1 \not\subseteq S^k_{\mathbf{P},C}$. Using again $T^k = S^k_{\mathbf{P},C}$, we find that $B^+_1\theta_1 \not\subseteq T^k$.

From $B^+_1\theta_1 \not\subseteq T^k$ and (50), we conclude that there is a rule $r_2 \in P_{k+1}$ with $\mathsf{sk}(r_2) = (B^+_2, B^-_2, H_2)$, a substitution θ_2, and a number $o < j$ such that

$$B^+_2\theta_2 \subseteq T^o_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F) \tag{54}$$

$$B^-_2\theta_2 \cap \mathcal{M} = \emptyset \tag{55}$$

$$H_2\theta_2 \cap B^+_1\theta_1 \not\subseteq T^o_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F) \tag{56}$$

By (52) and (53), there is $\alpha \in H_1\theta_1 \setminus T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$, and by (56), there is $\beta \in (H_2\theta_2 \cap B^+_1\theta_1) \setminus T^o_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$. Let $F'' := \mathcal{M} \setminus \{\alpha, \beta\}$. We claim that the set of facts $G' := \gamma_{F''}(F'')$ and substitution $\sigma' := \gamma_{F''}(\theta_1 \cup \theta_2)$ satisfy the conditions for $r_2 \xrightarrow{+}_C r_1$ in Definition 2. By definition, G' does not contain function symbols.

(P1) $B^+_2\sigma' \subseteq G'$ by (54) and $T^o_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F) \subseteq F''$. For the latter, note that $\beta \notin T^o_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$, and that $\alpha \notin T^o_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$ since $\alpha \notin T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$ and $T^o_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F) \subseteq T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$ (by $o < j$).

(P2) $B^-_2\sigma' \cap G' = \emptyset$ by (55).

(P3) $B^+_1\sigma' \subseteq G' \cup H_2\sigma'$ by (50) and $\alpha \notin T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$.

(P4) $B^-_1\sigma' \cap (G' \cup H_2\sigma') = \emptyset$ by (51) and $H_2\theta_2 \subseteq \mathcal{M}$.

(P5) $B^+_1\sigma' \not\subseteq G'$ by our choice of β.

(P6) $H_1\sigma' \not\subseteq G' \cup H_2\sigma'$ is a consequence of our choice of α and $\alpha \notin T^j_{\mathsf{GL}(P^{k+1}_1 \cup C, \mathcal{M})}(F)$.

Since $\mathcal{M} \models_{\mathsf{SM}} F \cup P \cup C$, we have $\bot \notin F''$ and, so, $G' \models C$. As a consequence, $r_2 \xrightarrow{+}_C r_1$. This contradicts the assumption that $r_1 \in P_1^k$ while $r_2 \in P_{k+1}$, and thus refutes our initial assumption that $S_{\mathbf{P},C}^{k+1} \not\subseteq T^{k+1}$. We have thus shown that $S_{\mathbf{P},C}^{k+1} \subseteq T^{k+1}$.

By $S_{\mathbf{P},C}^{k+1} \subseteq T^{k+1}$, we have $\bot \notin S_{\mathbf{P},C}^{k+1}$. The latter combined with Proposition 4 yields $S_{\mathbf{P},C}^{k+1} \models_{\mathsf{SM}} F \cup P_1^{k+1} \cup C$. Thus $S_{\mathbf{P},C}^{k+1} = T^{\infty}_{\mathsf{GL}(P_1^{k+1} \cup C, S_{\mathbf{P},C}^{k+1})}(F)$. Now since $S_{\mathbf{P},C}^{k+1} \subseteq T^{k+1} \subseteq \mathcal{M}$, we find that

$$\mathsf{GL}(P_1^{k+1} \cup C, S_{\mathbf{P},C}^{k+1}) \supseteq \mathsf{GL}(P_1^{k+1} \cup C, T^{k+1}) \supseteq \mathsf{GL}(P_1^{k+1} \cup C, \mathcal{M}).$$

So, $S_{\mathbf{P},C}^{k+1} = T^{\infty}_{\mathsf{GL}(P_1^{k+1} \cup C, S_{\mathbf{P},C}^{k+1})}(F) \supseteq T^{\infty}_{\mathsf{GL}(P_1^{k+1} \cup C, \mathcal{M})}(F) = T^{k+1}$ as required. □

Proof of Theorem 8. If $\bot \notin S_{\mathbf{P},C}^{n}$, then by Proposition 4, $S_{\mathbf{P},C}^{n} \models_{\mathsf{SM}} F \cup P \cup C$, which together with Lemma 6 implies that $S_{\mathbf{P},C}^{n}$ is the unique stable model.

If $\bot \in S_{\mathbf{P},C}^{n}$ assume for a contradiction that there exists set of facts $\mathcal{M}$, such that $\mathcal{M} \models_{\mathsf{SM}} F \cup P \cup C$. By Lemma 6, $\mathcal{M} = S_{\mathbf{P},C}^{n}$ which is not possible since no stable model may contain bottom. □

Theorems 2 and 5 can be generalised to programs that are R-acyclic and R-stratified under constraints:

Theorem 9. *For a set of facts F, a fact α, and a program P that is R-acyclic under a set of constraints C, deciding $P \cup F \cup C \models \alpha$ is* coN2ExpTime*-complete (*coNP*-complete) w.r.t. program (data) complexity. If P is also R-stratified under C, deciding $P \cup F \cup C \models \alpha$ becomes* 2ExpTime*-complete (*P*-complete) w.r.t. program (data) complexity.*

Proof. Since extending P with constraints does not increase the bound on the number of terms and checking satisfaction of constraints is in NP w.r.t. program complexity and polynomial w.r.t. data complexity, the claims follow from Theorems 2 and 5. □

8 Experimental Evaluation

In order to assess the practical utility of our solution, we conducted a case study with ChEBI. Our test datasets, software, and detailed results are published online.[1]

The ChEBI database (release 97) contains about 20,000 molecular structures and taxonomic relations for about 8,000 chemical classes, while the DL-based ontology contains taxonomic information only. To obtain rules for reasoning, we considered a sample of 500 molecules, with sizes ranging from 2 to 138 atoms. The structure of each molecule (given in *MDL Molfile* format) was converted to rules of the form (2). Chemical classes, such as *one-carbon molecule* or *organic hydroxy*, do not have machine-readable descriptions in ChEBI. We selected 50 chemical classes and manually formalised their human-readable descriptions as rules, such as (3) and (6). In addition, we defined 30 molecule classes that are characterised by small substructures (functional groups of 2 to 8 atoms), e.g., organic hydroxy. We modelled each with two rules of the form (8) and (9), using distinct predicates r and n for each pair of rules. Finally, existential quantifiers were skolemised, and conjunctions in rule heads were decomposed into multiple rules. This led to a program P with 78,957 rules, the largest of which had 38 body atoms (8 negative). P was not stratified, but was R-stratified and R-acyclic. In addition, we generated a set F of 530 facts of the form $C(a_C)$, one for each molecule or functional group. This allowed us to compute subsumptions between chemical classes: C is subsumed by C' iff $C'(a_C)$ is in the unique stable model of $P \cup F$.

We ran experiments on a desktop computer (2GHz quadcore CPU, 4GB RAM) running Linux. In a first experiment, we tried to compute a stable model of $P \cup F$ using DLV [Leone *et al.*, 2006], but the system failed to compute this result within a time limit of 600 seconds. In a second experiment, we split P into R-strata and consecutively computed the stable model of each stratum. Of the five R-strata of P, the first stratum P_1 contained 78,251 rules, while the 706 rules of the remaining four R-strata formed a stratified program P_2^5. We thus used DLV to compute the stable model of $P_1 \cup F$, converted the result into a new set of facts $S_{\mathbf{P}}^1$, and used DLV to compute the stable model of $S_{\mathbf{P}}^1 \cup P_2^5$. This took 17 seconds, with 13.5 seconds being used for actual reasoning in DLV.

We obtained 8,639 non-trivial subsumptions overall between chemical classes, (excluding reflexive subsumptions and subsumptions with auxiliary predicates), which we compared to ChEBI's manually created taxonomy. This revealed several omissions in ChEBI, e.g., the fact that every organic hydroxy (ChEBI id 33822) is an organooxygen compound (ChEBI id 36963), illustrating the practical relevance of our approach.

9 Related Work

Nonmonotonic extensions for existential rules are considered by Calì *et al.* [2009] using stratified negation, and more recently by Gottlob *et al.* [2012] using well-founded semantics. Another approach to nonmonotonic ontological modelling are $\mathcal{FDNC}$ programs [Eiter and Simkus, 2010], which are related to DLs and inherit many of their limitations in modelling finite structures.

Local stratification generalises stratification by considering the (infinite) groundings of normal logic programs [Przymusinski, 1989]. This condition is undecidable [Cholak and Blair, 1994], but does not generalise R-stratification as Exaple 8 shows.

Before providing the example, we formally define local stratification. Given a program P, a sequence of disjoint programs $\mathbf{P} = P_1, \ldots, P_n$ is a *local stratification* of P if $\mathsf{Ground}(P) = \bigcup_{i=1}^{n} P_i$ and, for all programs $P_i, P_j \in \mathbf{P}$, rules $(B_1^+, B_1^-, H_1) \in P_i$ and $(B_2^+, B_2^-, H_2) \in P_j$, and every atom $\alpha \in \mathsf{Pred}(H_1)$, we have: (i) if $\alpha \in \mathsf{Pred}(B_2^+)$ then $i \leq j$, and (ii) if $\alpha \in \mathsf{Pred}(B_2^-)$ then $i < j$. P is *locally stratified* if it has a local stratification.

[1] http://www.cs.ox.ac.uk/isg/people/despoina.magka/tools/ijcai13experiments.zip

Example 8. *Let r_1 and r_2 be defined as follows:*

$$r_1: \quad \mathsf{A}(x) \wedge \mathsf{S}(x,y) \wedge \mathbf{not}\ \mathsf{A}(y) \wedge \mathbf{not}\ \mathsf{B}(y) \rightarrow \mathsf{R}(x,y)$$
$$r_2: \quad \mathsf{A}(u) \wedge \mathsf{T}(u,v) \wedge \mathbf{not}\ \mathsf{R}(v,u) \rightarrow \mathsf{B}(u)$$

We have $r_1 \rightarrow r_2$ because there are no F and θ that satisfy (N1)–(N5) *for $r_1 \rightarrow r_2$, as this would require $\theta(u) = \theta(y)$, $\mathsf{A}(u)\theta \in F$ but $\mathsf{A}(y)\theta \notin F$; also, by $\mathsf{Pred}(H_1) \cap \mathsf{Pred}(B_2^+) = \emptyset$ it is $r_1 \not\rightarrow r_2$. Moreover, by $r_1 \not\rightarrow r_1$ and $r_2 \not\rightarrow r_2$, we derive that $P = \{r_1, r_2\}$ is R-stratified. P is not locally stratified as shown by the following instantiation of P.*

$$r_1': \quad \mathsf{A}(a) \wedge \mathsf{S}(a,b) \wedge \mathbf{not}\ \mathsf{A}(b) \wedge \mathbf{not}\ \mathsf{B}(b) \rightarrow \mathsf{R}(a,b)$$
$$r_2': \quad \mathsf{A}(b) \wedge T(b,a) \wedge \mathbf{not}\ \mathsf{R}(a,b) \rightarrow \mathsf{B}(b)$$

Since $\mathsf{B}(b)$ occurs both in the negative body of r_1' and in the head of r_2' and $\mathsf{R}(a,b)$ occurs both in the negative body of r_2' and in the head of r_1', it is not possible to define a local stratification for $\{r_1', r_2'\}$.

Further extensions along these lines led to *weak stratification* [Przymusinska and Przymunsinski, 1990], *effective stratification* [Bidoit and Froidevaux, 1991], *modular stratification* [Ross, 1994], and *left-to-right dynamic stratification* [Sagonas *et al.*, 2001], all of which are known or suspected to be undecidable in the presence of function symbols.

Many other works study the problem of recognising negation-free programs with finite models, e.g., Fagin *et al.* [2005], Krötzsch and Rudolph [2011] and Cuenca Grau *et al.* [2012]. Negation is rarely considered. *Omega-restrictedness* uses a kind of 'stratification' to ensure finiteness of stable models [Syrjänen, 2001]. Magka *et al.* [2012] define *semantic acyclicity* to ensure finite models in reasoning about structured objects but only consider stratified negation. In contrast, non-stratified negation is the key to our modelling solution in Section 3.

Any program can be partitioned in such a way that each problematic reliance cycle is fully contained in some part. One can then apply other acyclicity or stratification criteria to each part to extend our results to even wider program classes. Deutsch *et al.* [2008] apply a similar idea to acyclicity.

10 Conclusions

We showed that nonmonotonic existential rules under a stable model semantics can address complex real-world modelling problems, and presented novel conditions to ensure efficient, deterministic reasoning in these cases. Our experiments indicate that our approach can dramatically increase the performance of existing answer set programming engines, enabling them to address new, practically relevant application areas.

For future work, it is thus very promising to integrate our approach into existing rule engines, which will also allow more extensive evaluations. Section 7 suggests that cyclic or non-stratified programs could be 'repaired' by adding suitable constraints, which could inspire new tools for rule modelling. Equality theories often lead to additional reliances, whereas datatypes and numeric constraints could be exploited to discard reliances—further work is needed to study these effects.

Acknowledgements This work was supported by the Royal Society, the Seventh Framework Program (FP7) of the European Commission under Grant Agreement 318338, 'Optique', and the EPSRC projects ExODA, Score! and MaSI3.

References

[Apt and Bol, 1994] Krzysztof R. Apt and Roland N. Bol. Logic programming and negation: A survey. *J. Log. Program.*, 19/20:9–71, 1994.

[Baget *et al.*, 2011a] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654, 2011.

[Baget *et al.*, 2011b] Jean-François Baget, Marie-Laure Mugnier, and Michaël Thomazo. Towards farsighted dependencies for existential rules. In *Proc. 5th Int. Conf. on Web Reasoning and Rule Systems (RR'11)*, 2011.

[Bauer *et al.*, 1973] M. Bauer, D. Brand, M. Fischer, A. Meyer, and M. Paterson. A note on disjunctive form tautologies. *ACM SIGACT News*, 5(2):17–20, 1973.

[Bidoit and Froidevaux, 1991] Nicole Bidoit and Christine Froidevaux. Negation by default and unstratifiable logic programs. *Theor. Comput. Sci.*, 78(1):86–112, 1991.

[Calì *et al.*, 2009] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. In *PODS*, pages 77–86. ACM, 2009.

[Calì *et al.*, 2010] Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *LICS*, 2010.

[Calì *et al.*, 2012] Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

[Cholak and Blair, 1994] Peter Cholak and Howard A. Blair. The complexity of local stratification. *Fundam. Inform.*, 21(4):333–344, 1994.

[Cuenca Grau *et al.*, 2012] Bernardo Cuenca Grau, Ian Horrocks, Markus Krötzsch, Clemens Kupke, Despoina Magka, Boris Motik, and Zhe Wang. Acyclicity conditions and their application to query answering in description logics. In *KR*, 2012.

[Dantsin *et al.*, 2001] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, 2001.

[de Matos *et al.*, 2010] Paula de Matos, Rafael Alcántara, Adriano Dekker, Marcus Ennis, Janna Hastings, Kenneth Haug, Inmaculada Spiteri, Steve Turner, and Christoph Steinbeck. Chemical entities of biological interest: an update. *Nucleic Acids Research*, 38:249–254, 2010.

[Deutsch *et al.*, 2008] Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In *PODS*, 2008.

[Eiter and Simkus, 2010] Thomas Eiter and Mantas Simkus. FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM TOCL*, 11(2), 2010.

[Eiter *et al.*, 2012] Thomas Eiter, Thomas Krennwallner, Patrik Schneider, and Guohui Xiao. Uniform evaluation of nonmonotonic DL-programs. In *FoIKS*. Springer, 2012.

[Fagin *et al.*, 2005] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

[Ferreira and Couto, 2010] João D. Ferreira and Francisco M. Couto. Semantic similarity for automatic classification of chemical compounds. *PLoS Computational Biology*, 6(9), 2010.

[Gkoutos *et al.*, 2012] Georgios Gkoutos, Paul Schofield, and Robert Hoehndorf. Computational tools for comparative phenomics: the role and promise of ontologies. *Mammalian Genome*, 23(9–10):669–679, 2012.

[Gottlob *et al.*, 2012] Georg Gottlob, André Hernich, Clemens Kupke, and Thomas Lukasiewicz. Equality-friendly well-founded semantics and applications to description logics. In *AAAI*, 2012.

[Greco *et al.*, 2012] Sergio Greco, Francesca Spezzano, and Irina Trubitsyna. On the termination of logic programs with function symbols. In *ICLP (Tech. Comm.)*, 2012.

[Hastings *et al.*, 2012] Janna Hastings, Despoina Magka, Colin R. Batchelor, Lian Duan, Robert Stevens, Marcus Ennis, and Christoph Steinbeck. Structure-based classification and ontology in chemistry. *J. Cheminf.*, 4:8, 2012.

[Hastings *et al.*, 2013] Janna Hastings, Paula de Matos, Adriano Dekker, Marcus Ennis, Bhavana Harsha, Namrata Kale, Venkatesh Muthukrishnan, Gareth Owen, Steve Turner, Mark Williams, and Christoph Steinbeck. The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Research*, 41(Database-Issue):456–463, 2013.

[Krötzsch and Rudolph, 2011] Markus Krötzsch and Sebastian Rudolph. Extending decidable existential rules by joining acyclicity and guardedness. In *IJCAI*, pages 963–968, 2011.

[Leone *et al.*, 2006] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM TOCL*, 7(3), 2006.

[Magka *et al.*, 2012] Despoina Magka, Boris Motik, and Ian Horrocks. Modelling structured domains using description graphs and logic programming. In *ESWC*, 2012.

[Motik and Rosati, 2010] Boris Motik and Riccardo Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.

[Motik *et al.*, 2009] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, and Ulrike Sattler. Representing ontologies using description logics, description graphs, and rules. *Artif. Intell.*, 173(14), 2009.

[Mugnier, 2009] Marie-Laure Mugnier. Conceptual Graph Rules and Equivalent Rules: A Synthesis. In *ICCS*, pages 23–31, 2009.

[Przymusinska and Przymunsinski, 1990] H. Przymusinska and T. C. Przymunsinski. Weakly stratified logic programs. *Fundam. Inf.*, 13(1):51–65, March 1990.

[Przymusinski, 1989] Teodor C. Przymusinski. On the declarative and procedural semantics of logic programs. *J. Autom. Reasoning*, 5(2):167–205, 1989.

[Ross, 1994] Kenneth A. Ross. Modular stratification and magic sets for Datalog programs with negation. *J. ACM*, 41(6):1216–1266, 1994.

[Sagonas *et al.*, 2001] Konstantinos F. Sagonas, Terrance Swift, and David Scott Warren. The limits of fixed-order computation. *Theor. Comput. Sci.*, 254:465–499, 2001.

[Syrjänen, 2001] Tommi Syrjänen. Omega-restricted logic programs. In *LPNMR*, pages 267–279, 2001.