Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Finding Finite Herbrand Models

Stefan Borgwardt Barbara Morawska

LTCS-Report 11-04

Postal Address:
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
01062 Dresden

http://lat.inf.tu-dresden.de

Visiting Address:
Nöthnitzer Str. 46
Dresden

Finding Finite Herbrand Models

Stefan Borgwardt Barbara Morawska

Abstract

We show that finding finite Herbrand models for a restricted class of first-order clauses is ExpTime-complete. A Herbrand model is called finite if it interprets all predicates by finite subsets of the Herbrand universe. The restricted class of clauses consists of anti-Horn clauses with monadic predicates and terms constructed over unary function symbols and constants.

The decision procedure can be used as a new goal-oriented algorithm to solve linear language equations and unification problems in the description logic $\mathcal{FL}_0$. The new algorithm has only worst-case exponential runtime, in contrast to the previous one which was even best-case exponential.

1 Introduction

Satisfiability of formulas in First Order Logic (FOL) has always been of interest for computer science and is an active field of research. The main problem is that satisfiability of such formulas is not even semi-decidable. Thus, the focus lies on finding algorithms that decide satisfiability for restricted classes. A possible approach is to use restrictions on the resolution or superposition calculi to obtain decision procedures [7, 9].

Related to this is the problem of *model building* that asks for an actual model witnessing the satisfiability of the given clauses. Additionally, one usually asks for a finite representation of such a model. For example, the completeness proofs of resolution-style inference systems often explicitly construct (counter-)models, but there are also other approaches [2, 10, 15].

Here, we want to study a related problem. We not only want our models to be finitely representable, but actually ask for finite Herbrand models. We call a Herbrand model finite if each predicate is interpreted by a finite subset of the infinite Herbrand universe. This problem is semi-decidable since the set of all finite Herbrand interpretations over a fixed signature is recursively enumerable. Since this problem has not been studied before, it is unknown whether it is decidable for arbitrary first-order formulae. The existence of finite Herbrand

models implies the existence of finite models in the usual sense, where the domain is required to be finite, but the other implication does not hold in general.

We restrict ourselves to finite sets of *propagation rules*, which are anti-Horn clauses that use only monadic predicates and function symbols, one constant symbol, and one variable. In particular, we do not allow the equality predicate. These sets of clauses can be seen as skolemized versions of Ackermann formulas, for which satisfiability is known to be decidable [6, 9]. This class of clause sets is also similar to the decidable Bernays-Schönfinkel class [9], but neither is actually included in the other.

In this report, we show that the problem of deciding the existence of a finite Herbrand model for a finite set of propagation rules is ExpTime-complete. Our decision procedure is aided by a new computational model that we call *propagation nets*. The process of building a model is simulated by the process of saturating the net with terms. This process terminates iff a finite Herbrand model exists. We decide this by analyzing the structure of the net.

The problem of finding finite Herbrand models for a set of propagation rules occurred while designing a new unification procedure for the description logic $\mathcal{FL}_0$. The unification problem in this logic was shown to be ExpTime-complete in [1]. There, solving unification in $\mathcal{FL}_0$ is shown to be equivalent to solving linear language equations. It turns out that the problem of solving these equations reduces in a natural way to the problem of finding finite Herbrand models for propagation rules. In this reduction, variables become predicates and their finite interpretation in the Herbrand universe defines a solution to the original language equation.

Our decision procedure thus provides a new way to solve linear language equations. It is worst-case exponential, but there are cases in which our algorithm runs in polynomial time. Thus, it has advantages over the previous algorithm [1], which is always exponential. Moreover, our procedure can be modified to compute solutions for the formal language equations.

We think that this method of finding finite Herbrand models can find various applications and can be generalized to larger classes of clauses. As detailed above, it has an immediate application to unification and formal language equations.

In the next section, we introduce propagation rules and describe a simple flattening procedure. In Section 3, we will present propagation nets and illustrate their connection to propagation rules. After describing the ideas on how to find finite Herbrand models for finite sets of propagation rules, we will present a decision procedure in Section 4. There we prove its correctness and that it always terminates after at most exponential time. We also show that the problem is ExpTime-hard, which is done by a reduction from linear language equations in Section 5. In Section 6, we will summarize our main results and finally present our conclusions in Section 7.

2 Propagation Rules

We first introduce the clausal formalism for which we want to decide the existence of finite Herbrand models. We deal with a special kind of first-order anti-Horn clauses, which we call *propagation rules*. The signature of our first-order logic consists of a finite set $\mathcal{P}$ of unary predicates, a finite set $\mathcal{F}$ of unary function symbols, one constant a, and one variable x. The variable x is always implicitly universally quantified.

We will use the notation $P, Q, P_1, P_2, \ldots$ for predicates and $f, g, \ldots$ for function symbols. Every ground term over this signature is of the form $f_1(\ldots f_n(a)\ldots)$, which we will abbreviate as $f_1 \ldots f_n(a)$.

Definition 1. A *propagation rule* is a clause of the form

- $\top \to P_1(a) \vee \cdots \vee P_n(a)$ *(positive clause)*,
- $P_0(a) \to P_1(a) \vee \cdots \vee P_n(a)$, or
- $P_0(t_0) \to P_1(t_1) \vee \cdots \vee P_n(t_n)$

for $P_0, \ldots, P_n \in \mathcal{P}$ and non-ground terms $t_0, \ldots, t_n$ over $\mathcal{F}$ and x.[1]

We assume that the reader is familiar with the concept of Herbrand interpretations (see, e.g., [9]). In the following, a Herbrand interpretation $\mathcal{H}$ over the above signature is called *finite* if it interprets every predicate $P \in \mathcal{P}$ by a finite set $P^{\mathcal{H}}$. The task we are interested in is to decide the existence of finite Herbrand models for finite sets of propagation rules. As a first step towards a decision procedure, we will flatten the propagation rules to get rid of terms of depth larger than 0, except in some special clauses.

Definition 2. A finite set $\mathcal{C}$ of propagation rules is called *normalized* if there is a set $\mathcal{D}(\mathcal{C}) \subseteq \mathcal{P} \times \mathcal{F}$ such that

- For every $(P, f) \in \mathcal{D}(\mathcal{C})$, we have $P^f \in \mathcal{P}$ and the two clauses $P^f(x) \to P(f(x))$ (*increasing clause*) and $P(f(x)) \to P^f(x)$ (*decreasing clause*) are contained in $\mathcal{C}$.
- All clauses in $\mathcal{C}$ except the ones above must be *flat*, i.e., of the form
  1) $\top \to P_1(a) \vee \cdots \vee P_n(a)$,
  2) $P_0(a) \to P_1(a) \vee \cdots \vee P_n(a)$, or
  3) $P_0(x) \to P_1(x) \vee \cdots \vee P_n(x)$

[1] Note that n might be 0, in which case the right-hand side of the clause is $\bot$. Positive clauses must be ground since otherwise no finite Herbrand model could exist.

for $P_0, \ldots, P_n \in \mathcal{P}$.

We denote by $\mathcal{D}^f(\mathcal{C})$ the set $\{P \in \mathcal{P} \mid (P, f) \in \mathcal{D}(\mathcal{C})\}$ for any $f \in \mathcal{F}$.

The interesting property of such sets is that in order to check whether a flat clause $P_0(x) \to P_1(x) \vee \cdots \vee P_n(x)$ is satisfied by a term, one only needs to consider this term. The only case where different terms can occur in the same instance of a clause are the increasing and decreasing clauses, which only allow a very limited connection between the terms, i.e., adding and removing the leading function symbol.

The special predicates P^f represent all the terms in P that have the prefix f: For any Herbrand model $\mathcal{H}$ of the clauses $P^f(x) \to P(f(x))$ and $P(f(x)) \to P^f(x)$ and any word $w \in \mathcal{F}^*$, the term $f(w(a))$ is in $P^{\mathcal{H}}$ iff $w(a)$ is in $P^{f\,\mathcal{H}}$. These predicates are stored in the set $\mathcal{D}(\mathcal{C})$, which acts as an "interface" between terms of different lengths: A clause can only contain different terms if a predicate P^f with $(P, f) \in \mathcal{D}(\mathcal{C})$ is involved.

We assume in the following that $\mathcal{C}$ is a finite set of propagation rules and will transform it into a normalized set $\mathcal{C}'$. Observe that the ground clauses of $\mathcal{C}$ can remain unchanged since they are already flat by Definition 1. Thus, we only have to consider clauses where the variable x occurs in each atom.

1) We initialize $\mathcal{C}' := \mathcal{C}$ and the set $\mathcal{D}(\mathcal{C}')$ as follows.

   If for $(P, f) \in \mathcal{P} \times \mathcal{F}$ there is a unique predicate Q such that $Q(x) \to P(f(x))$ and $P(f(x)) \to Q(x)$ are in $\mathcal{C}'$ and the same does not hold for any other pair in $\mathcal{P} \times \mathcal{F}$ and Q, then we rename Q to P^f and add (P, f) to $\mathcal{D}(\mathcal{C}')$.[2] (The cases where Q is not unique or assumes the roles of both P_1^f and P_2^g for $P_1 \neq P_2$ or $f \neq g$ are handled by the next step.)

2) While $\mathcal{C}'$ is not yet normalized, we choose a clause $c = P_0(t_0) \to P_1(t_1) \vee \cdots \vee P_n(t_n)$ that still contains a term $t_i = f(t_i')$ for some $f \in \mathcal{F}$ and term t_i'.

   If (P_i, f) is already in $\mathcal{D}(\mathcal{C})$, we simply replace $P_i(t_i)$ by $P_i^f(t_i')$.

   Otherwise, we add the new predicate P_i^f to $\mathcal{P}$,[2] extend $\mathcal{D}(\mathcal{C}')$ by (P_i, f), add the clauses $P_i(f(x)) \to P_i^f(x)$ and $P_i^f(x) \to P_i(f(x))$ to $\mathcal{C}$, and replace the atom $P_i(t_i)$ in c by $P_i^f(t_i')$.

Since $\mathcal{C}$ is finite, the terms occurring in $\mathcal{C}$ have a maximal depth, which decreases by 1 in each step of the flattening procedure. Thus, this process terminates. In contrast to common flattening procedures for first-order clauses, we do not use new variables or equality [2].

[2] If this predicate already occurs in $\mathcal{C}'$, we rename it.

Lemma 3. *$\mathcal{C}$ has a finite Herbrand model iff $\mathcal{C}'$ does.*

Proof. Regarding step 1), the renaming of predicates does not affect the existence of a finite Herbrand model. For step 2), it is enough to prove the claim for one replacement step.

If $\mathcal{H}$ is a finite Herbrand model of $\mathcal{C}'$ before the replacement, we can extend it to the new predicate P_i^f as follows: $P_i^{f\,\mathcal{H}} := \{w(a) \mid f(w(a)) \in P_i^{\mathcal{H}}\}$. The set $P_i^{f\,\mathcal{H}}$ is finite since $P_i^{\mathcal{H}}$ is finite. Since the atom $P_i(f(w(a)))$ is true in $\mathcal{H}$ iff $P_i^f(w(a))$ is, this interpretation satisfies the increasing and decreasing clauses as well as all the clauses in which $P_i(f(t_i'))$ was replaced by $P_i^f(t_i')$.

Let now $\mathcal{H}$ be a finite Herbrand model of $\mathcal{C}'$ after the replacement. Since it satisfies the increasing and decreasing clauses for $(P_i, f) \in \mathcal{D}(\mathcal{C}')$, we again have $P_i^{f\,\mathcal{H}} = \{w(a) \mid f(w(a)) \in P_i^{\mathcal{H}}\}$. Thus, $P_i(f(w(a)))$ is true in $\mathcal{H}$ iff $P_i^f(w(a))$ is, which implies that changing $P_i^f(t_i')$ back to $P_i(f(t_i'))$ does not affect the truth value of the clause under $\mathcal{H}$. □

For each non-ground atom $P(f_1 \dots f_n(x))$ occurring in $\mathcal{C}$, polynomially many new predicates $P^{f_1}, P^{f_1 f_2}, \dots, P^{f_1 \cdots f_n}$ are created. Furthermore, for each of the new predicates, only two new clauses of polynomial size are created. Thus, the size of $\mathcal{C}'$ is polynomial in the size of $\mathcal{C}$.

Example 4. Consider the set

$$\begin{aligned}\mathcal{C}_1 := \{&\top \to P_0(a),\ P_0(f(x)) \to \bot,\ P_0(g(x)) \to \bot,\\ &P_3(a) \to \bot,\ P_3(f(x)) \to \bot,\ P_3(g(x)) \to P_0(x),\ P_0(x) \to P_3(g(x)),\\ &P_0(a) \to P_1(a),\ P_1(a) \to P_0(a),\\ &P_2(x) \to P_3(x) \vee P_1(f(x)),\ P_3(x) \to P_2(x),\ P_1(f(x)) \to P_2(x)\\ &P_1(x) \to P_2(x) \vee P_1(g(x)),\ P_2(x) \to P_1(x),\ P_1(g(x)) \to P_1(x)\}\end{aligned}$$

of propagation rules.

To construct the normalized set $\mathcal{C}_1'$, observe that P_0 can be renamed to P_3^g. Subsequently, the atoms $P_3(f(x))$, $P_1(f(x))$, $P_1(g(x))$, $P_3^g(f(x))$, and $P_3^g(g(x))$ are replaced by $P_3^f(x)$, $P_1^f(x)$, $P_1^g(x)$, $P_3^{gf}(x)$, and $P_3^{gg}(x)$, respectively. This results in the set

$$\mathcal{D}(\mathcal{C}_1') = \{(P_3, f), (P_3, g), (P_1, f), (P_1, g), (P_3^f, g), (P_3^g, g)\}$$

and the corresponding increasing and decreasing clauses. The resulting flat clauses are the following:

$$\begin{aligned}
&\top \to P_3^g(a),\ P_3^{gf}(x) \to \bot,\ P_3^{gg}(x) \to \bot,\\
&P_3(a) \to \bot,\ P_3^f(x) \to \bot,\\
&P_3^g(a) \to P_1(a),\ P_1(a) \to P_3^g(a),\\
&P_2(x) \to P_3(x) \vee P_1^f(x),\ P_3(x) \to P_2(x),\ P_1^f(x) \to P_2(x),\\
&P_1(x) \to P_2(x) \vee P_1^g(x),\ P_2(x) \to P_1(x),\ P_1^g(x) \to P_1(x).
\end{aligned}$$

We will use the set $\mathcal{C}_1'$ throughout this report to illustrate the presented constructions and algorithms.

A notion that will be useful later is that of *possibilities* of clauses and sets of clauses.

Definition 5. For a flat clause c, the set $\texttt{possibilities}(c)$ is the set of all predicates occurring on the right-hand side of c. For a finite set $\mathcal{C} = \{c_1, \ldots, c_n\}$ of flat clauses, we define $\texttt{possibilities}(\mathcal{C})$ as the set

$$\{\{P_1, \ldots, P_n\} \mid \forall i \in \{1, \ldots, n\} : P_i \in \texttt{possibilities}(c_i)\}.$$

For example, the clause $P_1(a) \to P_2(a) \vee P_3(a)$ has the possibilities P_2 and P_3, while the set $\{P_1(x) \to P_2(x) \vee P_3(x), \top \to P_0(a)\}$ has the possibilities $\{P_2, P_0\}$ and $\{P_3, P_0\}$.

To simplify the following descriptions, we assume that any normalized set $\mathcal{C}$ of propagation rules contains at most one positive clause, which is of the form $\top \to A(a)$, and that the special predicate A otherwise only occurs on the left-hand side of other ground clauses. If this is not the case, we can introduce a new predicate A, add the clause $\top \to A(a)$ to $\mathcal{C}$, and replace $\top$ by $A(a)$ in every other positive clause. It is easy to see that this modification does not affect the existence of a finite Herbrand model for $\mathcal{C}$.

Example 6. For the set $\mathcal{C}_1'$ from Example 4, we simply add $\top \to A(a)$ to $\mathcal{C}_1'$ and replace the propagation rule $\top \to P_3^g(a)$ by $A(a) \to P_3^g(a)$.

3 Propagation Nets

We now introduce a new computational model that will be used to decide the existence of finite Herbrand models for finite sets of propagation rules. These *propagation nets* are similar to Petri nets, hence we use notions borrowed from the theory of Petri nets, like places, transitions, tokens, markings, and firings, which will be fully explained below. In Section 3.2, we will show how to use

propagation nets to simulate the search for Herbrand models of finite sets of propagation rules.

A propagation net consists of places and transitions which are connected by directed arcs in such a way that places are connected only to transitions and transitions are connected only to places. A computation in this structure moves words from places to other places using the transitions between them. If a place has several outgoing arcs to transitions, it can choose one of them to *fire*. This means that a word from this place is transported to the transition and then distributed to all places reachable from this transition by outgoing arcs. An arc from a place to a transition can also change the word by adding a letter or removing the first letter. An arc from a transition to a place can filter out words that should not be transported to the place.

Actually, the firing of a transition does not remove the word from the place but just deactivates it. The goal is to find a computation of the propagation net thats starts with a given distribution of words among places and terminates in the sense that all words are deactivated.

Definition 7. A *propagation net* $\mathcal{N} = (P, T, \Sigma, E, I, \pi, \tau)$ consists of

- a finite set P of *places*,
- a finite set T of *transitions*,
- a finite alphabet Σ,
- a set $E \subseteq (P \times T) \cup (T \times P)$ of *arcs*,
- an *initial marking* $I : (P \cup T) \to \mathcal{P}(\Sigma^*)$ and $I_a : P \to \mathcal{P}(\Sigma^*)$,
- a partial *filter function* $\pi : \big(E \cap (T \times P)\big) \to \Sigma \cup \{\varepsilon\}$, and
- a *successor function* $\tau : \big(E \cap (P \times T)\big) \to \Sigma \cup \{f^{-1} | f \in \Sigma\} \cup \{\varepsilon\}$.

A *token* in $\mathcal{N}$ is a word over Σ. A *marking* M of $\mathcal{N}$ is a pair of mappings $M : (P \cup T) \to \mathcal{P}(\Sigma^*)$ and $M_a : P \to \mathcal{P}(\Sigma^*)$ assigning to each place and each transition finite sets of tokens such that $M_a(p) \subseteq M(p)$ for every $p \in P$. The set $M(p)$ is called the set of tokens of a place $p \in P$, while the set $M(t)$ is called the set of tokens of a transition $t \in T$ in the marking M. The set $M_a(p)$ is the set of *active* tokens of a place p in M. We always assume that the initial marking I is a proper marking, i.e., it satisfies the above conditions.

We say that a token w *matches* the filter $\pi(t, p)$ of an arc $(t, p) \in E \cap (T \times P)$ if either

(i) $\pi(t, p)$ is undefined (no restriction on w),

(ii) $\pi(t,p) = \varepsilon$ and then $w = \varepsilon$, or

(iii) $\pi(t,p) = f \in \Sigma$ and then w starts with f.

There are two elementary operations on markings:

- *Deactivating a token at a place*: A token w is deactivated at a place $p \in P$ by removing it from $M_a(p)$, if it is in $M_a(p)$, and adding it to $M(p)$, if it is not already in $M(p)$. Note that w need not be in $M(p)$ to be deactivated.
- *Producing a token at a transition*: A token w is produced at a transition $t \in T$ by adding it to $M(t)$. This operation has the side effect of also *producing* the token at all places $p \in P$ with $(t,p) \in E$. This secondary operation is executed only if w matches the filter $\pi(t,p)$. If this is the case and $w \notin M(p)$, then w is added to $M(p)$ and $M_a(p)$. Otherwise, the token w is not added to the marking at p.

A *firing* in $\mathcal{N}$ is a triple $\mathfrak{f} = (p, w, t)$, where p is a place, w is a token, and t is a transition such that $(p,t) \in E$ and $\tau(p,t)w$ is defined, i.e., if $\tau(p,t) = f^{-1}$, then w begins with f. The *result* of firing $\mathfrak{f}$ in a marking M is a new marking M' which is defined as follows:

1. Initialize $M' := M$ and $M'_a := M_a$.
2. Deactivate the token w at p in M'.
3. Compute the *successor token* $w' := \tau(p,t)w$.
4. Produce w' at t in M', thereby also producing w' at every place reachable from t by an outgoing arc whose filter matches w'.

If M' is the result of the firing $\mathfrak{f}$ in M, then we write $M \xrightarrow{\mathfrak{f}} M'$. If $M(p) = M'(p)$ for all $p \in P$, this firing is called *unproductive* in M; otherwise, it is called *productive*. An unproductive firing only removes an active token from the marking, while a productive firing also introduces new active tokens.

Given a marking M_0, a *firing sequence (starting in M_0)* is a finite sequence $M_0 \xrightarrow{\mathfrak{f}_1} \ldots \xrightarrow{\mathfrak{f}_m} M_m$ of firings. If the initial marking is not important or clear from the context, we also denote this sequence simply by $\mathfrak{f}_1, \ldots, \mathfrak{f}_m$. The marking M_m is called the *final marking* of this sequence. The sequence is called *terminating* if M_m is *stable*, i.e., $M_{m,a}(p) = \emptyset$ holds for all $p \in P$. We say that $\mathcal{N}$ *terminates* if it has a terminating firing sequence that starts in I. Note that such a firing sequence has to end with a nonproductive firing since otherwise new active tokens would be created.

Figures 1, 2, and 3 depict a simple propagation net and the effect of different firings on the initial marking.

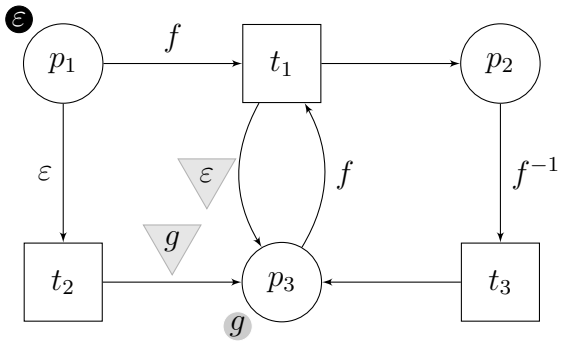

Figure 1: A simple propagation net with $P = \{p_1, p_2, p_3\}$ and $T = \{t_1, t_2, t_3\}$. Edge labels denote the functions π and τ, where filters are depicted as triangles. Filled circles are the tokens of the initial marking; active tokens have a black background.

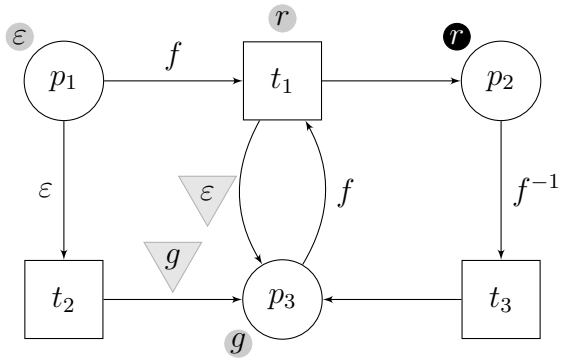

Figure 2: The propagation net from Figure 1 after firing (p_1, ε, t_1). The token f is produced at t_1 and p_2, but not at p_3 since f does not match the filter $\tau(t_1, p_3) = \varepsilon$.

3.1 Other Computational Models

We will now compare the behavior of propagation nets to that of traditional models of computation.

The terminology employed for defining propagation nets is borrowed from Petri nets [11, 12], although there are several key differences to ordinary Petri nets. In propagation nets, tokens are not atomic objects, but have the structure of words over an alphabet Σ. Additionally, transitions do not need to be synchronized, i.e., do not require the input token to be present at every input place.

In that respect, propagation nets behave much more like two-way alternating automata on finite words [4, 8, 3]. Places are the existential states, i.e., there is a nondeterministic choice between the output transitions. Transitions are the universal states since they distribute each token to all output places. Contrary to the standard model of automata, however, propagation nets do not read an input word, but rather write several words, i.e., the tokens that are produced.

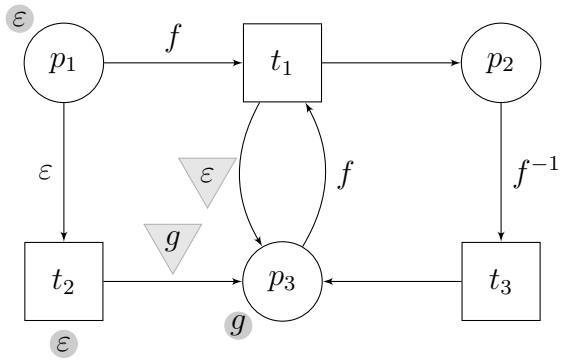

Figure 3: The propagation net from Figure 1 after firing (p_1, ε, t_2). The resulting marking is stable, which means that the propagation net terminates.

The computational model that comes closest to propagation nets is that of two-way alternating automata on finite trees [13, 5]. One can imagine the set Σ^* as a tree rooted in ε, where each node u has a successor fu for each $f \in \Sigma$.[3] In this representation, a finite subtree of Σ^* represents the set of all tokens produced by a terminating firing sequence. As described before, the places and transitions are existential and universal states, respectively. Adding the prefix f to a token w then corresponds to moving from w to fw in the tree, while keeping w or removing the first letter is similar to an ε- or a backwards-transition, respectively.

However, again there are some differences: The initial marking allows to designate arbitrary tokens as the starting point of the computation, while tree automata usually start only at the root ε, and only in one state. Furthermore, two-way alternating tree automata require all computations to end in a final state.[4] In propagation nets, however, coming back to a place with the same token is considered acceptable, because nonproductive firings are allowed, while this might lead to a cycle in the computation of an alternating automaton.

3.2 From Clauses to Propagation Nets

We now present a translation from a normalized set $\mathcal{C}$ of propagation rules to a propagation net $\mathcal{N}_\mathcal{C}$. The goal is to express the finite Herbrand models of $\mathcal{C}$ by stable markings of $\mathcal{N}_\mathcal{C}$.

We will represent terms by tokens, clauses by places, and predicates by transitions. From a clause, a token can be transferred to any of the predicates that occur in the clause. From a predicate, a token is then distributed to all clauses that have this predicate on their left-hand side. The filter function allows to dis-

[3]Usually, positions are interpreted from left to right. However, in our setting, new symbols are always concatenated on the left. This is because linear terms are built in this direction.

[4]Final states are basically transitions without outgoing arcs.

card those terms (tokens) that are irrelevant for satisfaction of the clause. The successor function allows to express increasing and decreasing clauses by adding or removing letters, respectively. For a flat clause, the successor function is ε, i.e., it leaves the term as it is. The initial marking simply consists of the active token ε at $\top \to A(a)$.

Definition 8. Let $\mathcal{C}$ be a normalized set of propagation rules over the set $\mathcal{P}$ of predicates and the set $\mathcal{F}$ of function symbols. The propagation net $\mathcal{N}_\mathcal{C} := (\mathcal{C}, \mathcal{P}, \mathcal{F}, E_\mathcal{C}, I_\mathcal{C}, \pi_\mathcal{C}, \tau_\mathcal{C})$ has the following components:

- $E_\mathcal{C} := \big\{(c, P_i) \mid c = \ldots \to P_1(t_1) \vee \cdots \vee P_n(t_n) \in \mathcal{C} \text{ and } i \in \{1, \ldots, n\}\big\}$
  $\cup \big\{(P_0, c) \mid c = P_0(t_0) \to \cdots \in \mathcal{C}\big\}$
- $I_\mathcal{C} : (\mathcal{C} \cup \mathcal{P}) \to \mathcal{P}(\mathcal{F}^*) : c \mapsto \begin{cases} \{\varepsilon\} & \text{if } c = \top \to A(a) \\ \emptyset & \text{otherwise} \end{cases}$
- $I_{\mathcal{C},a} := I_\mathcal{C}$
- $\pi_\mathcal{C}\big(P_0, P_0(t_0) \to \ldots\big) := \begin{cases} \varepsilon & \text{if } t_0 = a \\ \text{undefined} & \text{if } t_0 = x \\ f & \text{if } t_0 = f(x) \end{cases}$
- $\tau_\mathcal{C}\big(P_0(t_0) \to P_1(t_1) \vee \cdots \vee P_n(t_n), P_i\big) := \begin{cases} f & \text{if } t_0 = x,\ t_i = f(x) \\ f^{-1} & \text{if } t_0 = f(x),\ t_i = x \\ \varepsilon & \text{otherwise} \end{cases}$
- $\tau_\mathcal{C}(\top \to A(a), A) := \varepsilon$

Note that this means that a clause is connected to all its possibilities. Thus, every firing (c, w, P) of a clause c represents a possibility of c. Firing sequences can thus be seen as sequences of applying possibilities to tokens on the left-hand side of clauses: If $w(a)$ is a term in $P^\mathcal{H}$ for a Herbrand interpretation $\mathcal{H}$ and we want $\mathcal{H}$ to satisfy a clause $P(x) \to P_1(x) \vee \cdots \vee P_n(x)$, then we have to find a possibility P_i for which to put $w(a)$ into $P_i^\mathcal{H}$. In this way, propagation nets provide a convenient way to express sequences of applying possibilities in order to satisfy a set of clauses. These sequences always start with the token ε at the clause $\top \to A(a)$ since this is the only clause without precondition.

Example 9. Since the propagation net for the set $\mathcal{C}_1'$ from Example 4 is quite large, we consider instead the smaller set

$$\begin{aligned}\mathcal{C}_2 = \{&\top \to A(a), A(a) \to P_1(a) \vee P_2(a),\ P_2(x) \to P_1^f(x) \vee P_3(x),\\ &P_1^f(x) \to P_1(f(x)),\ P_1(f(x)) \to P_1^f(x)\}\end{aligned}$$

of propagation rules. The propagation net $\mathcal{N}_{\mathcal{C}_2}$ is depicted in Figure 4.

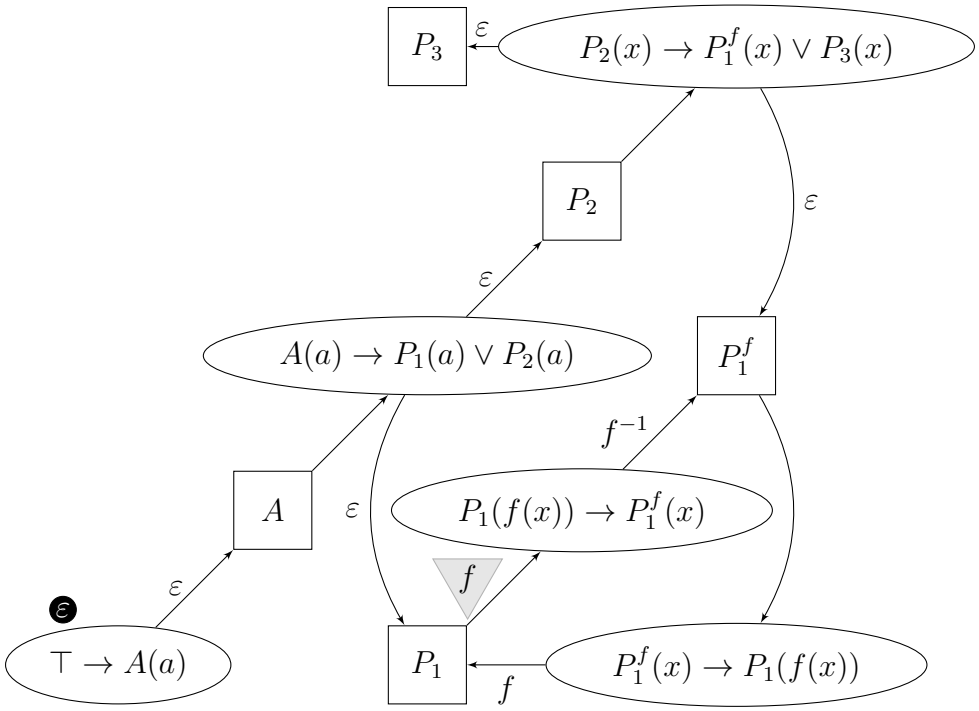

Figure 4: The propagation net for $\mathcal{C}_2$ from Example 9, with initial marking.

In the proof of the following lemma, we describe how this propagation net can be used to construct finite Herbrand models of $\mathcal{C}$.

Lemma 10. *$\mathcal{C}$ has a finite Herbrand model iff $\mathcal{N}_\mathcal{C}$ terminates.*

Proof. Given a finite Herbrand model $\mathcal{H}$ of $\mathcal{C}$, we inductively define a firing sequence of $\mathcal{N}_\mathcal{C}$ that satisfies the following invariant:

- For every predicate $P \in \mathcal{P}$ and token $w \in M_j(P)$ in any of the constructed markings, we have $w(a) \in P^\mathcal{H}$.

We will show that while there are active tokens in the marking M_j, we can find a firing that deactivates one of these tokens such that the invariant is satisfied in the resulting marking M_{j+1}.

We start the construction with $M_0 := I_\mathcal{C}$. Since the initial marking contains no tokens at transitions (i.e., predicates), the invariant is vacuously satisfied. Let now M_j be a marking of $\mathcal{N}_\mathcal{C}$ that satisfies the invariant and $c \in \mathcal{C}$ have an active token $w \in M_{j,a}(c)$. We consider the structure of c.

If $c = \top \to A(a)$, then c has no incoming arc, and thus w must already have been in $M_{0,a}(c) = I_\mathcal{C}(c)$. By definition of $I_\mathcal{C}$, we have $w = \varepsilon$. Since $\mathcal{H}$ satisfies c, we have $a \in A^\mathcal{H}$. This means that the firing (c, ε, A) preserves the invariant since only the successor token $\tau_\mathcal{C}(c, A)w = \varepsilon$ is added to $M_j(A)$.

Otherwise, c is of the form $P_0(t_0) \to P_1(t_1) \vee \cdots \vee P_n(t_n)$, where either $t_0 = t_1 = \cdots = t_n \in \{x, a\}$ or $n = 1$ and $\{t_0, t_1\} = \{x, f(x)\}$ for some $f \in \mathcal{F}$. In any case, by assumption we have $w \in M_j(P_0)$ since w cannot be active at c without having been produced at P_0 first. Thus, we have $w(a) \in P_0^{\mathcal{H}}$ by the invariant. Furthermore, w must match $\pi_{\mathcal{C}}(P_0, c)$ since otherwise it would not have been produced at c. Since $\mathcal{H}$ satisfies c, there is $i \in \{1, \ldots, n\}$ and a term $u(a)$ in $P_i^{\mathcal{H}}$ that depends on the cases for t_0 and t_i depicted in the following table:

t_0	t_i	$\pi_{\mathcal{C}}(P_0, c)$	$\tau_{\mathcal{C}}(c, P_i)$	w	u	$\tau_{\mathcal{C}}(c, P_i)w$
a	a	ε	ε	ε	ε	ε
x	x	undefined	ε	w	w	w
x	$f(x)$	undefined	f	w	fw	fw
$f(x)$	x	f	f^{-1}	fw'	w'	w'

In all of these cases, the firing (c, w, P_i) preserves the invariant since the successor token $\tau_{\mathcal{C}}(c, P_i)w$ is always equal to u and $u(a) \in P_i^{\mathcal{H}}$.

The invariant ensures that the sets $M_j(P)$ cannot grow indefinitely since their size is bounded by the size of the finite sets $P^{\mathcal{H}}$. Thus, there is an index $m_0 \in \mathbb{N}$ after which no more productive transitions are fired. The remaining active tokens are then deactivated by unproductive firings, resulting in a stable marking M_m.

Let now $M_0 \xrightarrow{\mathfrak{f}_1} \ldots \xrightarrow{\mathfrak{f}_m} M_m$ be a terminating firing sequence of $\mathcal{N}_{\mathcal{C}}$ starting in $M_0 = I_{\mathcal{C}}$. We define the finite Herbrand interpretation $\mathcal{H}$ by

$$P^{\mathcal{H}} := \{w(a) \mid \exists j \in \{1, \ldots, m\} : \mathfrak{f}_j = (c, w', P), w = \tau_{\mathcal{C}}(c, P)w'\}$$

for every predicate P. The set $P^{\mathcal{H}}$ thus contains $w(a)$ for all tokens w that were produced by the firing sequence at the transition P. This implies that for every P and $w(a) \in P^{\mathcal{H}}$ we have $w \in M_m(P)$ since every token that was produced at P must still be present in the final marking of P.

To show that $\mathcal{H}$ is a model of $\mathcal{C}$, consider any clause $c \in \mathcal{C}$. If $c = \top \to A(a)$, then $M_m(c) = M_0(c) = \{\varepsilon\}$ since this clause has no incoming transitions. Since the token ε is not active in M_m at c, the sequence must contain the firing (c, ε, A) that deactivated it. Since $\tau_{\mathcal{C}}(c, A) = \varepsilon$, the term $a = (\tau_{\mathcal{C}}(c, A)\varepsilon)\,(a)$ is contained in $A^{\mathcal{H}}$, and thus c is satisfied by $\mathcal{H}$.

Let now c be of the form $P_0(t_0) \to P_1(t_1) \vee \cdots \vee P_n(t_n)$, where either $t_0 = t_1 = \cdots = t_n \in \{x, a\}$ or $n = 1$ and $\{t_0, t_1\} = \{x, f(x)\}$ for some $f \in \mathcal{F}$. To show that $\mathcal{H}$ satisfies c, we assume that the left-hand side of the clause is satisfied by $\mathcal{H}$ with some term $w(a)$, i.e., $w(a) \in P_0^{\mathcal{H}}$ and $P_0(w(a))$ is an instance of $P_0(t_0)$. Since this means that w was produced at P_0, it also had to be distributed to all connected places with matching filter. In particular, c is such a place with $(P_0, c) \in E_{\mathcal{C}}$.

If w does not match the filter $\pi_{\mathcal{C}}(P_0, c)$ of this arc, then $P_0(w(a))$ is not an instance of $P_0(t_0)$, which contradicts our assumption. Otherwise, w was added to the marking of c, which implies that we have $w \in M_m(c)$ in the final marking.

Since M_m is stable, the token w is not active in M_m at c, and thus the firing sequence must contain a firing (c, w, P_i) for some $i \in \{1, \ldots, n\}$ that deactivated it.

- If $t_0 = t_i = a$, then $w = \varepsilon = \tau_{\mathcal{C}}(c, P_i)$, and thus $a = (\tau_{\mathcal{C}}(c, P_i)w)(a) \in P_i^{\mathcal{H}}$.
- If $t_0 = f(x)$ and $t_i = x$, then $w = fw'$ for some $w' \in \mathcal{F}^*$. Thus, $w'(a) = (f^{-1}w)(a) = (\tau_{\mathcal{C}}(c, P_i)w)(a) \in P_i^{\mathcal{H}}$.
- If $t_0 = x$ and $t_i = f(x)$, then $fw(a) = (\tau_{\mathcal{C}}(c, P_i)w)(a) \in P_i^{\mathcal{H}}$.

In each of the above cases, the ground instance of c that has $P_0(w(a))$ on its left-hand side is satisfied by $\mathcal{H}$. □

Example 11. Consider the propagation net $\mathcal{N}_1 := \mathcal{N}_{\mathcal{C}_1'}$ based on the rules from Example 4. If we omit unproductive firings, the following is a terminating firing sequence of $\mathcal{N}_1$ starting in $I_{\mathcal{C}_1'}$:

$$
\begin{aligned}
&(\top \to A(a), \varepsilon, A),\ (A(a) \to P_3^g(a), \varepsilon, P_3^g),\\
&(P_3^g(x) \to P_3(g(x)), \varepsilon, P_3),\ (P_3(x) \to P_2(x), g, P_2),\\
&(P_2(x) \to P_1(x), g, P_1),\ (P_1(g(x)) \to P_1^g(x), g, P_1^g),\\
&(P_1^g(x) \to P_1(x), \varepsilon, P_1)
\end{aligned}
$$

Since this representation is hard to understand, in the following we abbreviate firings like $(P_1(x) \to P_2(x) \vee P_1^g(x), g, P_2)$ by $P_1(g) \to P_2(g)$ and join "adjacent" firings that share a ground atom:

$$
\begin{array}{ccccccc}
 & & & & P_3(g) & \longrightarrow & P_2(g) \longrightarrow P_1(g) \\
 & & & \nearrow & & & \searrow \\
\top & \longrightarrow & A(\varepsilon) & \longrightarrow & P_3^g(\varepsilon) & & P_1^g(\varepsilon) \longrightarrow P_1(\varepsilon)
\end{array}
$$

In this representation, the missing nonproductive firings would appear as arrows leading "back" to a previously created atom, thus forming a cycle in the graph. It is easy to read off the corresponding finite Herbrand model $\mathcal{H}$ of $\mathcal{C}_1'$ (see the proof of Lemma 10):

$$
\begin{aligned}
A^{\mathcal{H}} &= P_1^{g\mathcal{H}} = P_3^{g\mathcal{H}} = \{a\},\\
P_1^{\mathcal{H}} &= \{a, g(a)\},\\
P_2^{\mathcal{H}} &= P_3^{\mathcal{H}} = \{g(a)\},\\
P_1^{f\mathcal{H}} &= P_3^{f\mathcal{H}} = P_3^{gf\mathcal{H}} = P_3^{gg\mathcal{H}} = \emptyset.
\end{aligned}
$$

3.3 Behavior of Propagation Nets

Our goal is not to decide termination of arbitrary propagation nets, but only of those of the form $\mathcal{N}_\mathcal{C}$ for normalized sets $\mathcal{C}$ of propagation rules. In essence, we will present a decision procedure for the existence of finite Herbrand models for these clause sets. However, we will use propagation nets to formulate the ideas behind the decision procedure and to prove its correctness. Before we present the algorithm, we take a more detailed look at propagation nets of the form $\mathcal{N}_\mathcal{C}$.

Termination of Propagation Nets

We first analyze what it means for $\mathcal{N}_\mathcal{C}$ to have a terminating firing sequence starting in $I_\mathcal{C}$. A first observation is that the initial marking $I_\mathcal{C}$ only causes the token ε to be produced at A. Any such firing sequence will thus start with one active token ε and gradually distribute it to other predicates, while sometimes increasing it. There are two reasons why this might not be possible. First, it may be impossible to avoid a contradiction, i.e., a clause with $\bot$ on the right-hand side, in any firing sequence starting in $I_\mathcal{C}$. The other possibility is that every firing sequence that avoids all contradictions is forced into a cycle of creating ever longer tokens. Thus, in order to terminate, the length of the produced tokens has to be bounded.

To analyze the detailed structure of terminating firing sequences, we introduce the notion of *replacement sequences*.

Definition 12. Let $P \in \mathcal{X} \subseteq \mathcal{P}$ and $w = fw' \in \mathcal{F}^+$. A $(P, \mathcal{X}, w)$-*replacement sequence* is a firing sequence of $\mathcal{N}_\mathcal{C}$ starting in M_0 and ending in M_m such that

- M_0 only contains the token w at P and the active token w at all clauses with $P(x)$ or $P(f(x))$ on the left-hand side,
- M_m only contains tokens with the suffix w,
- $w \in M_m(Q)$ iff $Q \in \mathcal{X}$, and
- if $w' \in M_{m,a}(c)$, then $w' = w$ and $c = Q(f(x)) \to Q^f(x)$.

A (P, ε)-*replacement sequence* is a firing sequence starting in M_0 and ending in M_m such that

- M_0 only contains the token ε at P and the active token ε at all clauses with $P(x)$ or $P(a)$ on the left-hand side, and
- M_m is stable.

The *height* of a replacement sequence is the maximal number $|w'| - |w|$ for any token w' in M_m.

It is easy to see that any terminating firing sequence starting in $I_{\mathcal{C}}$ consists of the firing $(\top \to A(a), \varepsilon, A)$ and an (A, ε)-replacement sequence. Thus, our goal is to decide the existence of such replacement sequences.

If there is an (A, ε)-replacement sequence of height 0, then only the token ε is produced in this sequence. Deciding the existence of such sequences is easy, as will be demonstrated later (see Algorithm 2). But first, we assume that we have to look for (A, ε)-replacement sequences of height larger than 0.

If such a sequence produces a token $w = fw' \neq \varepsilon$ at a predicate P, this means that w must have been active in all clauses with $P(x)$ or $P(f(x))$ on the left-hand side at some point in the sequence. We can now extract a $(P, \mathcal{X}, w)$-replacement sequence as follows: Starting from the active token w at all clauses with $P(x)$ or $P(f(x))$ on the left-hand side, we extract all firings that deactivate these tokens and the tokens produced from these firings, except firings of the form $(Q(f(x)) \to Q^f(x), w, Q^f)$. The extracted firings form the replacement sequence and the set $\mathcal{X}$ consists of all those predicates Q at which w was produced in this sequence.

Assume that this replacement sequence has height h. If a longer token w' is produced in this sequence at a predicate Q, we can use the same procedure to extract a $(Q, \mathcal{Y}, w')$-replacement sequence, which will be of height smaller than h. This process can be continued until the height of the replacement sequences is 0.

Example 13. Consider the terminating firing sequence from Example 11. It consists of the firing $(\top \to A(a), \varepsilon, A)$, followed by an (A, ε)-replacement sequence. The firing $(P_3^g(x) \to P_3(g(x)), \varepsilon, P_3)$ produces the token g at all clauses with $P_3(x)$ or $P_3(g(x))$ on the left-hand side, which is the starting point of a replacement sequence.

The corresponding $(P_3, \{P_3, P_2, P_1\}, g)$-replacement sequence is

$$(P_3(x) \to P_2(x), g, P_2),\ (P_2(x) \to P_1(x), g, P_1),$$
$$(P_2(x) \to P_3(x) \vee P_1^f(x), g, P_3),\ (P_1(x) \to P_2(x) \vee P_1^g(x), g, P_2).$$

Note that the last two firings are unproductive.

In this way, every terminating firing sequence can be decomposed into nested replacement sequences. To decide termination of $\mathcal{N}_{\mathcal{C}}$, we will iteratively construct all possible replacement sequences, starting with those of height 0. These will then be used to build replacement sequences of increasingly larger heights, until we can construct an (A, ε)-replacement sequence.

In the following, we will describe the intuition behind the algorithm which will be presented in Section 4.

Replacement Sequences of Height 0

To construct replacement sequences of height 0 for a given predicate P and a word w, we use the notion of possibilities introduced in Section 2. We define the set $\texttt{possibilities}(P)$ to contain all possibilities of the set of all flat clauses with $P(x)$ on the left-hand side. Such a possibility represents one way of firing all these flat clauses by specifying which predicates will contain the token w afterwards. Of course, one possibility $\{Q_1, \ldots, Q_n\}$ for P only expresses the beginning of the desired replacement sequence. We then also have to consider the possibilities for each of the reached predicates $Q_1, \ldots, Q_n$ and repeat this process until no new predicates are reached.

Since we want to construct only replacement sequences of height 0, we must prevent this process to reach predicates of the form P^f with $(P, f) \in \mathcal{D}(\mathcal{C})$. Thus, we define $\texttt{possibilities}(P^f(x) \to P(f(x))) := \emptyset$ and extend the set $\texttt{possibilities}(P^f)$ to also consider this increasing clause. Thus, we have that $\texttt{possibilities}(P^f) = \emptyset$, which indicates that we have no way of dealing with the token w if it is produced at P^f.[5]

Example 14. We can use the possibilities of the predicates to construct the $(P_3, \{P_3, P_2, P_1\}, g)$-replacement sequence of height 0 from Example 13 as follows:

- For P_3, we have the possibility $\{P_2\}$, which yields the firing $(P_3(x) \to P_2(x), g, P_2)$, which creates the token g at P_2.
- P_2 has the possibilities $\{P_1, P_3\}$ and $\{P_1, P_1^f\}$. The first one yields the firings $(P_2(x) \to P_1(x), g, P_1)$ and $(P_2(x) \to P_3(x) \vee P_1^f(x), g, P_3)$. The second possibility would lead to the active token g at P_1^f, and thus to the active token fg at P_1, which would force the replacement sequence to be of height > 0.
- Finally, the possibilities for P_1 are $\{P_2\}$ and $\{P_1^g\}$. We choose the first one, which yields the firing $(P_1(x) \to P_2(x) \vee P_1^g(x), g, P_2)$.

The token g is now deactivated at every place except at the decreasing clauses $P_3(g(x)) \to P_3^g(x)$ and $P_1(g(x)) \to P_1^g(x)$.

It is easy to see that any $(P, \mathcal{X}, w)$-replacement sequence can be changed into a $(P, \mathcal{X}, w')$-replacement sequence by substituting the suffix w by w' in every token that occurs in the sequence. Thus, the token w is not necessary to describe

[5] This restriction will be relaxed later.

the replacement sequence. Similarly, it is not important which firings are used to deactivate tokens, only which predicates are reached. We thus propose the notion of *shortcuts* to simplify the representation.

Definition 15. A *shortcut* is a pair $(P, \mathcal{X})$ with $P \in \mathcal{X} \subseteq \mathcal{P}$.

Of course, we are only interested in shortcuts $(P, \mathcal{X})$ for which a $(P, \mathcal{X}, w)$-replacement sequence exists. Since there may be several possibilities for each predicate P, there are several replacement sequences for P, and thus several shortcuts $(P, \mathcal{X}_1)$, $(P, \mathcal{X}_2)$, ... representing them. In Section 4, we will present an algorithm that computes all these shortcuts.

Example 16. Consider again the set $\mathcal{C}_1'$ of propagation rules from Example 4. The $(P_3, \{P_3, P_2, P_1\}, g)$-replacement sequence shown in Example 13 yields the shortcut $(P_3, \{P_3, P_2, P_1\})$. Similarly, we can find replacement sequences for P_1 and P_2, represented by the shortcuts $(P_1, \{P_1, P_2, P_3\})$ and $(P_2, \{P_1, P_2, P_3\})$.

Note that a token at P_2 cannot be produced by an increasing clause since neither (P_2, f) nor (P_2, g) are in $\mathcal{D}(\mathcal{C})$. Tokens at P_2 can only be produced by firing the clauses $P_3(x) \to P_2(x)$, $P_1^f(x) \to P_2(x)$, or $P_1(x) \to P_2(x) \vee P_1^g(x)$. Thus, every replacement sequence for P_2 will always be part of a replacement sequence for another predicate. This demonstrates that our algorithm does not need to compute shortcuts for predicates Q that have no pair (Q, f) in $\mathcal{D}(\mathcal{C})$.

Replacement Sequences of Larger Height

If we have shortcuts for all possible replacement sequences of height 0, we can use these as building blocks for replacement sequences of height 1 as follows. Such a sequence will contain firings of increasing clauses $P^f(x) \to P(f(x))$ w.r.t. some token w. This firing produces the token fw at all clauses having $P(x)$ or $P(f(x))$ on the left-hand side. This is a possible starting point for a $(P, \mathcal{X}, fw)$-replacement sequence of height 0.

If we have already computed a shortcut $(P, \mathcal{X})$, there is a firing sequence that deactivates the token fw and distributes it to all predicates of $\mathcal{X}$. This leaves us to consider the tokens that were created by this sequence at decreasing clauses. These clauses must be of the form $Q(f(x)) \to Q^f(x)$ for $Q \in \mathcal{X}$ since the token begins with f and is distributed only to predicates in $\mathcal{X}$. We then simply fire these decreasing clauses, which gets us back to the original token w.

Thus, when looking for replacement sequences of height 1, we can use the previously created shortcuts as possibilities for the predicates P^f. Recall that so far, we have defined these possibilities to be $\emptyset$. Now, each shortcut $(P, \mathcal{X})$ yields a possibility $\{Q^f \mid Q \in \mathcal{X} \cap \mathcal{D}^f(\mathcal{C})\}$ for the increasing clause $P^f(x) \to P(f(x))$. Hence, if there is no such shortcut, $\texttt{possibilities}(P^f) = \emptyset$ as before. However,

if there is at least one shortcut $(P, \mathcal{X})$, then $\texttt{possibilities}(P^f)$ can now be non-empty. With this new definition of `possibilities`, we can compute shortcuts for replacement sequences of height 1. These yield more possibilities and serve as building blocks for shortcuts representing replacement sequences of height 2, and so on.

The following procedure implements the computation of all possibilities for a predicate P w.r.t. a set $\mathcal{R}$ of previously computed shortcuts.

Algorithm 1 ($\texttt{possibilities}(\mathcal{C}, \mathcal{R}, P)$)**.**

Input: a normalized set $\mathcal{C}$ of propagation rules, a set $\mathcal{R}$ of shortcuts, and a predicate P
Output: the set of possibilities for P w.r.t. $\mathcal{C}$ and $\mathcal{R}$
  if $P = Q^f$ with $(Q, f) \in \mathcal{D}(\mathcal{C})$ **then**
    $\mathcal{L} \leftarrow \{\{Q_1^f, \ldots, Q_n^f\} \mid (Q, \mathcal{X}) \in \mathcal{R},\ \{Q_1, \ldots, Q_n\} = \mathcal{X} \cap \mathcal{D}^f(\mathcal{C})\}$
  else $\mathcal{L} \leftarrow \{\emptyset\}$
  for all $P(x) \rightarrow P_1(x) \vee \cdots \vee P_n(x) \in \mathcal{C}$ **do**
    $\mathcal{L} \leftarrow \{\mathcal{Y} \cup \{P_l\} \mid \mathcal{Y} \in \mathcal{L},\ l \in \{1, \ldots, n\}\}$
  return $\mathcal{L}$

As detailed before, the set $\texttt{possibilities}(\mathcal{C}, \mathcal{R}, P)$ represents the choices we can make when constructing a firing sequence that produces a token at P. If $P = Q^f$, then we have to fire the increasing clause $Q^f(x) \rightarrow Q(f(x))$, which is simulated by one of the available shortcuts $(Q, \mathcal{X}) \in \mathcal{R}$. Every such shortcut gives rise to one initial possibility. Subsequently, for each flat clause starting with P, we can choose among its possibilities. The combination of all these choices leads to several possible sets of reachable predicates.

Example 17. If the shortcuts $\mathcal{R} = \{(P_1, \{P_1, P_2, P_3\}), (P_3, \{P_1, P_2, P_3\})\}$ from Example 16 are available, we have

$$\begin{aligned}
&\texttt{possibilities}(\mathcal{C}_1', \mathcal{R}, P_1^f) = \{\{P_1^f, P_3^f, P_2\}\},\\
&\texttt{possibilities}(\mathcal{C}_1', \mathcal{R}, P_1^g) = \{\{P_1^g, P_3^g, P_1\}\},\\
&\texttt{possibilities}(\mathcal{C}_1', \mathcal{R}, P_3^f) = \emptyset, \text{ and}\\
&\texttt{possibilities}(\mathcal{C}_1', \mathcal{R}, P_3^g) = \{\{P_1^g, P_3^g\}\}.
\end{aligned}$$

Without the shortcuts, all these sets would be empty.

It remains to decide whether we can actually construct an (A, ε)-replacement sequence from the computed shortcuts.

Replacement Sequences for ε

To construct a replacement sequence for ε, we can use the same approach as above, only that we also have to consider the ground clauses of $\mathcal{C}$ since they

can also distribute ε to other predicates. Since we will not use this replacement sequence as a building block for other replacement sequences and only have to decide its existence, we will not compute shortcuts, but simply a set of *good* predicates.

Definition 18. A predicate $P \in \mathcal{P}$ is *good* if there is a (P, ε)-replacement sequence. All other predicates are *bad*.

Our goal can thus be reformulated as follows: We want to decide whether A is good. To do this, we inductively construct the set $\mathcal{B}$ of all bad predicates using the following procedure and then test whether $A \notin \mathcal{B}$. This approach is similar to the emptiness test for looping automata on infinite trees [14].

Initially, $\mathcal{B}$ is empty. We then saturate $\mathcal{B}$ according to the following rule: If all possibilities of the set

$$\mathcal{C}_P := \{c \in \mathcal{C} \mid c = P(x) \to \dots \text{ or } c = P(a) \to \dots\}$$

contain a predicate from $\mathcal{B}$, then we add P to $\mathcal{B}$.

Thus, if every possibility of firing all clauses beginning with P leads to a bad predicate, then P must also be bad. Note that the possibilities of $\mathcal{C}_P$ might include the possibilities for an increasing clause, which are defined based on the shortcuts computed so far. Thus, a predicate P^f without a shortcut $(P, \mathcal{X})$ is immediately bad. Similarly, a predicate P with a clause $P(x) \to \bot$ or $P(a) \to \bot$ in $\mathcal{C}$ will be recognized as bad.

Example 19. Consider the set $\mathcal{C}_1'$ from Example 4 and assume that we have not yet computed any shortcuts. The predicates P_1^f, P_1^g, P_3^f, P_3^g, P_3^{gf}, and P_3^{gg} are immediately bad. Because of the clause $A(a) \to P_3^g(a)$, A is also bad. Thus, there is no (A, ε)-replacement sequence of height 0.

We now consider the first shortcuts computed in Example 16. Since the predicate P_3^g is no longer bad, A is good. This means that there is an (A, ε)-replacement sequence of height 1. Observe that the one that was already shown in Example 11 uses the shortcut $(P_3, \{P_1, P_2, P_3\})$ to take care of the token ε at P_3^g.

The following procedure implements this computation and will be used in the main algorithm to decide if A is *good* w.r.t. a set $\mathcal{R}$ of available shortcuts.

Algorithm 2 (`isTerminating`$(\mathcal{C}, \mathcal{R})$)**.**

Input: a normalized set $\mathcal{C}$ of propagation rules and a set $\mathcal{R}$ of shortcuts
Output: `true` iff A is good w.r.t. $\mathcal{R}$

$\mathcal{B}_0 \leftarrow \emptyset$, $k \leftarrow 0$
repeat
  $\mathcal{B}_{k+1} \leftarrow \mathcal{B}_k$
    $\cup \{P \in \mathcal{P} \mid \exists P(x) \to P_1(x) \vee \cdots \vee P_n(x) \in \mathcal{C} : \{P_1, \ldots, P_n\} \subseteq \mathcal{B}_k\}$
    $\cup \{P \in \mathcal{P} \mid \exists P(a) \to P_1(a) \vee \cdots \vee P_n(a) \in \mathcal{C} : \{P_1, \ldots, P_n\} \subseteq \mathcal{B}_k\}$
    $\cup \{P^f \in \mathcal{P} \mid (P, f) \in \mathcal{D}(\mathcal{C}),\ \forall (P, \mathcal{X}) \in \mathcal{R}\ \exists Q \in \mathcal{X} \cap \mathcal{D}^f(\mathcal{C}) : Q^f \in \mathcal{B}_k\}$
  $k \leftarrow k + 1$
until $\mathcal{B}_k = \mathcal{B}_{k-1}$
if $A \in \mathcal{B}_k$ **then**
  return `false`
else
  return `true`

4 Deciding Termination

Considering all the observations about the behavior of $\mathcal{N}_\mathcal{C}$, we can now formulate our main algorithm that decides its termination.

Algorithm 3 (Main algorithm)**.**

Input: a normalized set $\mathcal{C}$ of propagation rules
Output: `true` iff $\mathcal{N}_\mathcal{C}$ terminates

$\mathcal{R}_0 \leftarrow \emptyset$, $i \leftarrow 0$
repeat
  if `isTerminating`$(\mathcal{C}, \mathcal{R}_i)$ **then return** `true`
  $\mathcal{R}_{i+1} \leftarrow$ `nextShortcuts`$(\mathcal{C}, \mathcal{R}_i)$
  $i \leftarrow i + 1$
until $\mathcal{R}_i = \mathcal{R}_{i-1}$
return `false`

The algorithm iteratively computes shortcuts representing replacement sequences of increasing height until it has found all of them. The sets $\mathcal{R}_i$ are used to store all shortcuts computed so far. In each iteration, the algorithm checks whether these shortcuts already suffice to prove termination of $\mathcal{N}_\mathcal{C}$ using `isTerminating`$(\mathcal{C}, \mathcal{R}_i)$ (Algorithm 2). If not, shortcuts for the next height are computed. If there are no

new shortcuts, the algorithm stops and returns `false`, indicating that $\mathcal{N}_\mathcal{C}$ does not terminate.

The procedure `nextShortcuts`$(\mathcal{C}, \mathcal{R})$ implements the computation of the shortcuts representing replacement sequences of the next height.

Algorithm 4 (`nextShortcuts`$(\mathcal{C}, \mathcal{R})$)**.**

Input: a normalized set $\mathcal{C}$ of propagation rules and a set $\mathcal{R}$ of shortcuts
Output: a set $\mathcal{R}'$ of shortcuts for the next height
  $\mathcal{T} \leftarrow \{(P, \{P\}, \emptyset) \mid r \in \mathcal{F},\ (P, r) \in \mathcal{D}(\mathcal{C})\}$
  while there is $(P, R_P, V_P) \in \mathcal{T}$ with $R_P \setminus V_P \neq \emptyset$ **do**
    $\mathcal{T} \leftarrow \mathcal{T} \setminus \{(P, R_P, V_P)\}$
    choose Q from $R_P \setminus V_P$
    for all $\mathcal{Y} \in$ `possibilities`$(\mathcal{C}, \mathcal{R}, Q)$ **do**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{(P, R_P \cup \mathcal{Y}, V_P \cup \{Q\})\}$
  return $\{(P, R_P) \mid (P, R_P, R_P) \in \mathcal{T}\}$

It uses the possibilities of the predicates to construct all possible shortcuts. Starting from an initial predicate P, it considers its possibilities and adds the reached predicates to the shortcut. This process is iterated until no more predicates are reached.

In this computation, we use a set $\mathcal{T}$ of triples of the form (P, R_P, V_P), where R_P is the set of predicates we have reached so far starting from P, and V_P contains those predicates of R_P that were already *visited*, i.e., for which we have considered all possibilities. Visiting a predicate Q thus corresponds to firing all clauses starting with $Q(x)$.

The computation of shortcuts for P starts with the triple $(P, \{P\}, \emptyset)$. In each step, we choose a triple $(P, R_P, V_P) \in \mathcal{T}$ that still contains an unvisited predicate $Q \in R_P \setminus V_P$ and consider the possibilities for Q. For each possibility $\mathcal{Y} \in$ `possibilities`$(\mathcal{C}, \mathcal{R}, Q)$, we add the new triple $(P, R_P \cup \mathcal{Y}, V_P \cup \{Q\})$ to $\mathcal{T}$ since the predicates from $\mathcal{Y}$ have been reached and Q has just been visited. The original triple is then removed from $\mathcal{T}$.

We continue this process until there are no more triples with unvisited predicates. A triple (P, R_P, R_P) then yields the shortcut (P, R_P). Note that no predicate Q with `possibilities`$(\mathcal{C}, \mathcal{R}, Q) = \emptyset$ can be in R_P since visiting Q only removes the current triple and does not create new ones. As mentioned before, we restrict the starting triples $(P, \{P\}, \emptyset)$ to satisfy that $(P, f) \in \mathcal{D}(\mathcal{C})$ for some $f \in \mathcal{F}$, since our algorithm only needs replacement sequences for such predicates.

Example 20. Consider the set $\mathcal{C}_1'$ from Example 4. We describe the computation of `nextShortcuts`$(\mathcal{C}_1', \emptyset)$ to see how the first shortcuts mentioned in Example 16 are actually computed. The idea behind the computation was already illus-

trated in Example 14. We start with the four triples $(P_1, \{P_1\}, \emptyset)$, $(P_3, \{P_3\}, \emptyset)$, $(P_3^f, \{P_3^f\}, \emptyset)$, and $(P_3^g, \{P_3^g\}, \emptyset)$.

Consider the expansion of the first triple, which first visits P_1. The possibilities for P_1 are $\{P_2\}$ and $\{P_1^g\}$ and yield the two triples $(P_1, \{P_1, P_2\}, \{P_1\})$ and $(P_1, \{P_1, P_1^g\}, \{P_1\})$. However, the set of possibilities for P_1^g is empty since there is no shortcut $(P_1, \mathcal{X})$, and thus the second triple is removed without being replaced by another one. The possibilities for P_2 are $\{P_3, P_1\}$ and $\{P_1^f, P_1\}$. The second resulting triple is again removed subsequently, which leaves the triple $(P_1, \{P_1, P_2, P_3\}, \{P_1, P_2\})$. Finally, P_3 is visited, which reaches no new predicates and thus results in the triple $(P_1, \{P_1, P_2, P_3\}, \{P_1, P_2, P_3\})$.

The second initial triple is expanded similarly. First, P_3 is visited, which has the only possibility $\{P_2\}$. This yields the triple $(P_3, \{P_3, P_2\}, \{P_3\})$. Again, visiting P_2 creates two triples, one of which is discarded. The remaining triple is $(P_3, \{P_3, P_2, P_1\}, \{P_3, P_2\})$, which is again expanded according to the possibilities $\{P_2\}$ and $\{P_1^g\}$ for P_1. We get two triples, $(P_3, \{P_3, P_2, P_1\}, \{P_3, P_2, P_1\})$ and $(P_3, \{P_3, P_2, P_1, P_1^g\}, \{P_3, P_2, P_1\})$. Since the set of possibilities for P_1^g is empty, the second triple is removed without being replaced by another one.

Since the predicate P_3 has no shortcuts, the sets of possibilities for P_3^f and P_3^g are empty, and thus the triples $(P_3^f, \{P_3^f\}, \emptyset)$ and $(P_3^g, \{P_3^g\}, \emptyset)$ are also removed.

To summarize, this computation yields the two shortcuts $(P_1, \{P_1, P_2, P_3\})$ and $(P_3, \{P_1, P_2, P_3\})$. As mentioned before, the shortcut $(P_3, \{P_3, P_2, P_1\})$ corresponds to the $(P_3, \{P_1, P_2, P_3\}, g)$-replacement sequence shown in Example 13.

4.1 Shortcuts and Replacement Sequences

In this section, we formalize the intuition behind the computed shortcuts, i.e., that they represent replacement sequences. More precisely, shortcuts computed in the i-th iteration of the main loop of Algorithm 3 represent all replacement sequences of height at most $i - 1$.

Lemma 21. *Let $i \geq 1$ be such that $\mathcal{R}_i$ was computed by Algorithm 3, $(P, \mathcal{X}) \in \mathcal{R}_i$, and $w \in \mathcal{F}^+$. Then there is a $(P, \mathcal{X}, w)$-replacement sequence of height $\leq i - 1$.*

Proof. Let $i \geq 1$ and assume that the claim holds for all indices i' with $i > i' \geq 1$. Since $(P, \mathcal{X}) \in \mathcal{R}_i$, this shortcut had to be computed by `nextShortcuts`$(\mathcal{C}, \mathcal{R}_{i-1})$. Thus, after the main loop of this algorithm, the set $\mathcal{T}$ must have contained the triple $(P, \mathcal{X}, \mathcal{X})$. By following the computation of this triple from $(P, \{P\}, \emptyset)$ to $(P, \mathcal{X}, \mathcal{X})$, we will construct the desired firing sequence starting in the marking M_0. This marking contains only the token w at P and the active token w at all clauses with $P(x)$ or $P(f(x))$ on the left-hand side. For the following, let $f \in \mathcal{F}$ be the first letter of w.

At any point in this construction, we will maintain the invariant that if we have reached the triple (P, R_P, V_P) in the computation of $(P, \mathcal{X}, \mathcal{X})$, then the current marking

- contains the token w at a predicate Q iff $Q \in R_P$,
- contains active tokens w at all clauses $Q(x) \to \ldots$ with $Q \in R_P \setminus V_P$ and all decreasing clauses $Q(f(x)) \to Q^f(x)$ with $Q \in R_P \cap \mathcal{D}^f(\mathcal{C})$,
- contains no more active tokens, and
- contains only tokens that have the suffix w and are of length $\leq |w| + (i-1)$.

The invariant is satisfied initially since the initial triple is $(P, \{P\}, \emptyset)$ and the only tokens in M_0 are w at P and active tokens w at all clauses with $P(x)$ or $P(f(x))$ on the left-hand side.

Assume that we have constructed a partial firing sequence $\mathfrak{f}_1 \ldots \mathfrak{f}_k$, the resulting marking satisfies the invariant, and the computation of $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_{i-1})$ has reached the triple (P, R_P, V_P). At some point, the main loop will choose this triple and a predicate $P' \in R_P \setminus V_P$. One of the possibilities for P' will be added to R_P, while P' is added to V_P. Since the computation leads to the triple $(P, \mathcal{X}, \mathcal{X})$, this means that there is a possibility $\mathcal{Y} \in \texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-1}, P')$ with $\mathcal{Y} \subseteq \mathcal{X}$. We now consider the computation of this set in $\texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-1}, P')$ in order to determine how to deactivate the token w at the flat and increasing clauses with $P'(x)$ on the left-hand side.

If P' is of the form Q^g with $(Q, g) \in \mathcal{D}(\mathcal{C})$, then we must deactivate the token w at the increasing clause $Q^g(x) \to Q(g(x))$. For this, we use the firing $(Q^g(x) \to Q(g(x)), w, Q)$, which produces the token gw at all clauses with $Q(x)$ or $Q(g(x))$ on the left-hand side. Fortunately, $\mathcal{Y}$ must contain a set $\{Q_1^g, \ldots, Q_n^g\}$ with $(Q, \mathcal{Z}) \in \mathcal{R}_{i-1}$ and $\{Q_1, \ldots, Q_n\} = \mathcal{Z} \cap \mathcal{D}^g(\mathcal{C})$. This implies that $\mathcal{R}_{i-1}$ is not empty, and thus $i - 1 \geq 1$. By induction, there is a $(Q, \mathcal{Z}, gw)$-replacement sequence $\mathfrak{f}'_1 \ldots \mathfrak{f}'_o$. This sequence results only in active tokens gw at the decreasing clauses $Q_j(g(x)) \to Q_j^g(x)$ for all $j \in \{1, \ldots, n\}$. All tokens produced by this sequence have the suffix gw (and thus the suffix w) and are of length $\leq |gw| + (i-2) = |w| + (i-1)$.

After concatenating $\mathfrak{f}'_1 \ldots \mathfrak{f}'_o$ to our partial firing sequence, we only have to take care of the active tokens gw at decreasing clauses. To do this, we append the firings $(Q_j(g(x)) \to Q_j^g(x), gw, Q_j^g)$ for every $j \in \{1, \ldots, n\}$. These firings produce the token w at all clauses with $Q_1^g, \ldots, Q_n^g$ on the left-hand side. If $Q_j^g \in \mathcal{Y}$ is already in V_P, then w has already been deactivated at all the clauses with $Q_j^g(x)$ on the left-hand side; otherwise, these tokens are still active. In any case, the invariant is still satisfied after appending all these firings to the partial firing sequence constructed so far.

This leaves us to deal with the active token w at the flat clauses of the form $P'(x) \to P'_1(x) \vee \cdots \vee P'_n(x)$. Since $\mathcal{Y}$ is an element of the set computed by $\texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-1}, P')$, at some point this procedure must have considered this clause and chosen some P'_l to be added to $\mathcal{Y}$. Thus, by adding the firing $(P'(x) \to P'_1(x) \vee \cdots \vee P'_n(x), w, P'_l)$ we can be certain that the invariant is still satisfied. The token w will be active at every clause having $P'_l(x)$ on the left-hand side iff $P'_l \notin V_P$.

After we have considered all $P' \in \mathcal{X}$, we have deactivated all tokens except the token w at the decreasing clauses $P'(f(x)) \to P'^f(x)$ with $P' \in \mathcal{X} \cap \mathcal{D}^f(\mathcal{C})$. Furthermore, all tokens occurring in this sequence have the suffix w. Thus, the constructed sequence is a $(P, \mathcal{X})$-replacement sequence. Since all tokens in the final marking are of length $\leq |w|+(i-1)$, the height of this sequence is $\leq i-1$. □

To show the correspondence in the other direction, we first show an auxiliary result that demonstrates how to extract replacement sequences of height $\leq i-1$ from a replacement sequence of height i. This formalizes the idea of the nested replacement sequences illustrated in Example 13.

Lemma 22. *Let $\mathfrak{f}_1, \ldots, \mathfrak{f}_m$ be a $(P, \mathcal{X}, w)$-replacement sequence of height i and $\mathfrak{f}_j = (Q^g(x) \to Q(g(x)), w, Q)$ for a $j \in \{1, \ldots, m\}$. Then there is a $(Q, \mathcal{Y}, gw)$-replacement sequence of height $\leq i-1$ that contains only firings from $\{\mathfrak{f}_1, \ldots, \mathfrak{f}_m\}$.*

Proof. We will extract a subsequence of the sequence $M_0 \xrightarrow{\mathfrak{f}_1} \ldots \xrightarrow{\mathfrak{f}_m} M_m$ that has the desired properties. We will start this sequence in the marking M'_0 that contains only the token gw at Q and the active token gw at all clauses with $Q(x)$ or $Q(g(x))$ on the left-hand side. Since the original firing sequence contained the firing $\mathfrak{f}_j$ and the subsequence will use only firings from $\{\mathfrak{f}_1, \ldots, \mathfrak{f}_m\}$, every token occurring in this subsequence is also contained in M_m.

Assume now that we have already constructed a partial firing sequence $\mathfrak{f}'_1 \ldots \mathfrak{f}'_{o'}$ with $\{\mathfrak{f}'_1, \ldots, \mathfrak{f}'_o\} \subseteq \{\mathfrak{f}_1, \ldots, \mathfrak{f}_m\}$ and that the current marking contains an active token w' that is either $\neq gw$ or at a place other than $Q(g(x)) \to Q^g(x)$ for $(Q, g) \in \mathcal{D}(\mathcal{C})$. This token is also contained in M_m and it is longer than w. Thus, there must be a firing $\mathfrak{f}_k$ that deactivates it in the original replacement sequence. This firing will produce either the same token at another predicate or a longer token. Thus, by adding $\mathfrak{f}_k$ to the partial firing sequence, we maintain the property that all tokens have the suffix gw.

After we have thus deactivated all unwanted active tokens, we have extracted a firing sequence that ends in a marking M'_o. Since we only used firings of the original firing sequence, M'_o contains only tokens from M_m. Since the new sequence starts with the token gw, which is longer than w, the height of this sequence is smaller than that of the original sequence. We define the set $\mathcal{Y}$ to contain all predicates Q' for which $gw \in M'_o(Q')$. The constructed sequence is a $(Q, \mathcal{Y}, gw)$-replacement sequence of $\mathcal{N}_\mathcal{C}$. □

We now prove the complementary result to Lemma 21: Every replacement sequence of $\mathcal{N}_\mathcal{C}$ of height at most i corresponds to a shortcut computed in the $i+1$-th iteration of the main algorithm. However, this shortcut does not need to have the same set $\mathcal{X}$ of reached predicates, but only a subset of it.

The reason for this is that a replacement sequence is not restricted to firings that deactivate tokens that are already present. Firings can always be applied, regardless of whether they are necessary to deactivate some token or not. This means that replacement sequences might contain irrelevant firings. However, Algorithm 3 computes shortcuts in such a way that only necessary firings are considered, i.e., only possibilities for predicates that were already reached.

Lemma 23. *Consider the variant of Algorithm 3 that never returns, but simply computes the sets $\mathcal{R}_i$ for all $i \geq 0$. Let $P \in \mathcal{D}^f(\mathcal{C})$. If there is a $(P, \mathcal{X}, fw)$-replacement sequence of height $\leq i$, then $(P, \mathcal{X}') \in \mathcal{R}_{i+1}$ for some $\mathcal{X}' \subseteq \mathcal{X}$.*

Proof. Let $i \geq 0$ and asume that the claim is true for all i' with $i > i' \geq 0$. Let $\mathfrak{f}_1 \dots \mathfrak{f}_m$ be a $(P, \mathcal{X}, fw)$-replacement sequence of $\mathcal{N}_\mathcal{C}$ of height $\leq i$ starting in M_0 and ending in M_m.

To show the claim, we will demonstrate that $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_i)$ will compute a triple $(P, R_P, R_P) \in \mathcal{T}$ with $R_P \subseteq \mathcal{X}$. Initially, $\mathcal{T}$ will contain the triple $(P, \{P\}, \emptyset)$ since we assumed that $P \in \mathcal{D}^f(\mathcal{C})$. We will follow this triple throughout the execution of the algorithm and maintain the following invariant: If the currently computed triple is (P, R'_P, V_P), then M_m contains the token fw at every $Q \in R'_P$. The invariant is satisfied initially since $\mathfrak{f}_1 \dots \mathfrak{f}_m$ is a $(P, \mathcal{X}, fw)$-replacement sequence, and thus contains the token fw at P by definition.

Let (P, R'_P, V_P) be the triple computed so far and assume that it satisfies the invariant. If $R'_P \setminus V_P$ is non-empty, the algorithm will at some point choose a predicate $P' \in R'_P \setminus V_P$ and consider the set $\texttt{possibilities}(\mathcal{C}, \mathcal{R}_i, P')$. In order to maintain the invariant, we have to show that there is a possibility $\mathcal{Y} \in \texttt{possibilities}(\mathcal{C}, \mathcal{R}_i, P')$ such that M_m contains fw at all predicates in this set.

We first consider the case that P' is of the form Q^g. By the invariant, we know that $M_m(Q^g(x) \to Q(g(x)))$ contains the token fw. Thus, there must be a firing $\mathfrak{f}_j = (Q^g(x) \to Q(g(x)), fw, Q)$. By Lemma 22, there is a set $\mathcal{Z} \subseteq \mathcal{P}$ such that there is a $(Q, \mathcal{Z}, gfw)$-replacement sequence of height smaller than i, and thus $\leq i-1$. Since this height cannot be negative, we have $i-1 \geq 0$. By assumption, we know that $\mathcal{R}_{i-1}$ must contain a shortcut $(Q, \mathcal{Z}')$ for some $\mathcal{Z}' \subseteq \mathcal{Z}$.

Thus, the initial set $\mathcal{L}$ computed by $\texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-1}, P')$ before the main loop will contain the set $\{Q^g_1, \dots, Q^g_n\}$ for $\{Q_1, \dots, Q_n\} = \mathcal{Z}' \cap \mathcal{D}^g(\mathcal{C})$. Since the $(Q, \mathcal{Z}, gfw)$-replacement sequence we have obtained by Lemma 22 is a subsequence of $\mathfrak{f}_1 \dots \mathfrak{f}_m$, the original sequence must contain a firing $(Q_j(g(x)) \to Q^g_j(x), gfw, Z^g_j)$ for each $Q_j \in \mathcal{Z}' \cap \mathcal{D}^g(\mathcal{C}) \subseteq \mathcal{Z}$. Thus, the final marking M_m contains the token fw at all of the predicates $Q^g_1, \dots, Q^g_n$.

It remains to be verified that this property is maintained in the main loop of $\texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-1}, P')$. For this, consider any clause $P'(x) \to P_1(x) \vee \cdots \vee P_n(x) \in \mathcal{C}$. Since M_m contains the token fw at P', there must be a firing $\mathfrak{f}_j = (P'(x) \to P_1(x) \vee \cdots \vee P_n(x), fw, P_l)$ with $j \in \{1, \ldots, m\}$ and $l \in \{1, \ldots, n\}$. Thus, by adding P_l to the set constructed in $\mathcal{L}$ so far, we do not violate the property that this set contains only predicates that are marked by fw in M_m.

Thus, $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_i)$ computes a triple (P, R_P, R_P) such that $fw \in M_m(P')$ for each $P' \in R_P$. This means that the set $\mathcal{R}_i$ contains the shortcut (P, R_P) with $R_P \subseteq \mathcal{X}$. □

In the following sections, we use these results to show that the main algorithm is sound and complete and terminates after time at most exponential in the size of $\mathcal{C}$.

4.2 Correctness

We now employ Lemma 21 to actually construct a terminating firing sequence of $\mathcal{N}_\mathcal{C}$ starting in $I_\mathcal{C}$.

Lemma 24. *(Soundness) If Algorithm 3 returns* `true`*, then* $\mathcal{N}_\mathcal{C}$ *terminates.*

Proof. Let $m \in \mathbb{N}$ be the index for which $\texttt{isTerminating}(\mathcal{C}, \mathcal{R}_m)$ returned `true` and $t \in \mathbb{N}$ be the index of the last set $\mathcal{B}_t$ computed by this algorithm, i.e., for which $\mathcal{B}_{t+1} = \mathcal{B}_t$ holds. First note that since $\mathcal{B}_t$ is equal to $\mathcal{B}_{t+1}$, we know that $P \in \mathcal{P} \setminus \mathcal{B}_t$ implies that

- for all clauses $P(x) \to P_1(x) \vee \cdots \vee P_n(x) \in \mathcal{C}$ there must be an index $l \in \{1, \ldots, n\}$ such that $P_l \notin \mathcal{B}_t$,
- for all clauses $P(a) \to P_1(a) \vee \cdots \vee P_n(a) \in \mathcal{C}$ there must be an index $l \in \{1, \ldots, n\}$ such that $P_l \notin \mathcal{B}_t$, and
- if $P = Q^f$ for $(Q, f) \in \mathcal{D}(\mathcal{C})$, there is a shortcut $(Q, \mathcal{Y}) \in \mathcal{R}_m$ such that ${Q'}^f \notin \mathcal{B}_t$ for every $Q' \in \mathcal{Y} \cap \mathcal{D}^f(\mathcal{C})$.

In order to construct a terminating firing sequence of $\mathcal{N}_\mathcal{C}$ starting in $I_\mathcal{C}$, we employ the same strategy as in the proof of Lemma 21. We will maintain the invariant that the current marking

- contains the token ε only at predicates that are not contained in $\mathcal{B}_t$ and
- does not contain active tokens other than ε.

We start the construction by deactivating the token ε at $\top \to A(a)$ as follows: Since $\texttt{isTerminating}(\mathcal{C}, \mathcal{R}_m)$ returned `true`, we know that $A \notin \mathcal{B}_t$. Thus, the firing $(\top \to A(a), \varepsilon, A)$ does not violate the invariant.

Assume now that we have already constructed a partial firing sequence $\mathfrak{f}_1 \dots \mathfrak{f}_m$ and the resulting marking satisfies the invariant. Let $c \in \mathcal{C}$ be a clause at which the token ε is active.

We first consider the case that c is of the form $P(t) \to P_1(t) \vee \dots \vee P_n(t)$, where t is either x or a. By the invariant we know that $P \notin \mathcal{B}_t$, and thus there is $l \in \{1, \dots, n\}$ with $P_l \notin \mathcal{B}_t$. Thus, by adding the firing (c, ε, P_l) we deactivate the active token ε at c and produce ε only at a predicate not contained in $\mathcal{B}_t$.

If c is of the form $P^f(x) \to P(f(x))$, then $(P, f) \in \mathcal{D}(\mathcal{C})$ and $P^f \notin \mathcal{B}_t$, and thus there is a shortcut $(P, \mathcal{X}) \in \mathcal{R}_m$ with $Q^f \notin \mathcal{B}_t$ for all $Q \in \mathcal{X} \cap \mathcal{D}^f(\mathcal{C})$. By firing $(P^f(x) \to P(f(x)), \varepsilon, P)$, the token f is produced at all clauses with $P(x)$ or $P(f(x))$ on the left-hand side. These tokens might be active, and thus we have to deactivate them.

By Lemma 21, there is an $(P, \mathcal{X}, f)$-replacement sequence $\mathfrak{f}'_1 \dots \mathfrak{f}'_o$ that results only in the active tokens f at the places $Q(f(x)) \to Q^f(x)$ with $Q \in \mathcal{X} \cap \mathcal{D}^f(\mathcal{C})$. After concatenating $\mathfrak{f}'_1 \dots \mathfrak{f}'_o$ to our partial firing sequence, we thus only have to take care of the possibly active tokens f at these decreasing clauses. To do this, we append the firing $(Q(f(x)) \to Q^f(x), f, Q^f)$ for every $Q \in \mathcal{X} \cap \mathcal{D}^f(\mathcal{C})$. These firings produce only tokens ε for predicates not contained in $\mathcal{B}_t$. Thus, the resulting sequence still satisfies the invariant.

Since there are only finitely many predicates, at some point all produced tokens ε have been deactivated in the constructed sequence. Since all tokens longer than ε are also deactivated, the sequence is terminating. □

The following lemma uses Lemma 23 to show that the algorithm computes sufficiently many shortcuts to prove termination of $\mathcal{N}_\mathcal{C}$.

Lemma 25. *(Completeness) If $\mathcal{N}_\mathcal{C}$ terminates, then Algorithm 3 returns* `true`*.*

Proof. Let $\mathfrak{f}_1 \dots \mathfrak{f}_m$ be a terminating firing sequence of $\mathcal{N}_\mathcal{C}$ starting in $I_\mathcal{C}$ and ending in M_m. Let further $i \in \mathbb{N}$ be the maximal length of tokens appearing in M_m. We consider the variant of Algorithm 3 that never returns and take a closer look at the execution of $\texttt{isTerminating}(\mathcal{C}, \mathcal{R}_i)$. Let $\mathcal{B}_t$ be the last of the sets computed by the algorithm. We claim that for every predicate $P \in \mathcal{P}$ with $\varepsilon \in M_m(P)$ we have $P \notin \mathcal{B}_t$.

Assume the converse, i.e., that there is a predicate $P \in \mathcal{B}_t$ with $\varepsilon \in M_m(P)$. Then there is an index $0 < j < t$ with $P \in \mathcal{B}_j$ and $P \notin \mathcal{B}_{j-1}$. We further assume that P is one of the first of these predicates in the sense that j is the smallest index such that $\mathcal{B}_j$ contains a predicate Q with $\varepsilon \in M_m(Q)$. There are two possible reasons for P to be included in $\mathcal{B}_j$.

1. There is a clause $P(t) \to P_1(t) \vee \cdots \vee P_n(t) \in \mathcal{C}$ with $t \in \{a, x\}$ such that $\{P_1, \ldots, P_n\} \subseteq \mathcal{B}_{j-1}$. By assumption, we know that $\varepsilon \notin M_m(P_l)$ for every $l \in \{1, \ldots, n\}$. Thus, the token ε must still be active in M_m at P, which contradicts the stability of M_m.

2. P is of the form Q^f and for every shortcut $(Q, \mathcal{Y}) \in \mathcal{R}_i$ there is a variable $Q' \in \mathcal{Y} \cap \mathcal{D}^f(\mathcal{C})$ with $Q'^f \in \mathcal{B}_{j-1}$.

   Since $\varepsilon \in M_m(Q^f)$, there must be a firing $\mathfrak{f}_j = (Q^f(x) \to Q(f(x)), \varepsilon, Q)$. Using the ideas from the proof of Lemma 22, it is easy to extract a subsequence of $\mathfrak{f}_1 \ldots \mathfrak{f}_m$ that is a $(Q, \mathcal{Y}', f)$-replacement sequence of height $\leq i - 1$ for some $\mathcal{Y}' \subseteq \mathcal{P}$.

   By Lemma 23, $\mathcal{R}_i$ contains a shortcut $(Q, \mathcal{Y}'')$ with $\mathcal{Y}'' \subseteq \mathcal{Y}'$. Thus, there is $Q' \in \mathcal{Y}'' \cap \mathcal{D}^f(\mathcal{C}) \subseteq \mathcal{Y}'$ with $Q'^f \in \mathcal{B}_{j-1}$. Since the $(Q, \mathcal{Y}', f)$-replacement sequence constructed above is a subsequence of $\mathfrak{f}_1 \ldots \mathfrak{f}_m$, the token f is also produced at Q' in the sequence $\mathfrak{f}_1 \ldots \mathfrak{f}_m$. Thus, it must still be contained in the final marking M_m at Q'. Since M_m is stable, the firing $(Q'(f(x)) \to Q'^f(x), f, Q^f)$ must occur in the sequence, and thus $\varepsilon \in M_m(Q'^f)$. However, since $Q'^f \in \mathcal{B}_{j-1}$, this contradicts the minimality of j.

This concludes the proof of the claim that any predicate P with $\varepsilon \in M_m(P)$ cannot be contained in $\mathcal{B}_t$.

Since the token ε at $\top \to A(a)$ cannot be active in M_m, it must have been deactivated by the firing $(\top \to A(a), \varepsilon, A)$. This means that $\varepsilon \in M_m(A)$, which implies $A \notin \mathcal{B}_t$. Thus, `isTerminating`$(\mathcal{C}, \mathcal{R}_i)$ returns `true`.

Assume that the unmodified version of Algorithm 3 returns `false`. Then there must be an index $j < i$ with $\mathcal{R}_j = \mathcal{R}_{j+1}$. Assume that j is the smallest such index. Since the result of `nextShortcuts` depends only on the previous shortcuts, the sets $\mathcal{R}_j, \mathcal{R}_{j+1}, \ldots, \mathcal{R}_i, \ldots$ are all equal. Since the result of `isTerminating` also depends only on the given shortcuts, we have `isTerminating`$(\mathcal{C}, \mathcal{R}_j) =$ `isTerminating`$(\mathcal{C}, \mathcal{R}_i) =$ `true`, contradicting the assumption that the main algorithm returns `false`. □

This concludes the proof of correctness of Algorithm 3.

4.3 Complexity

To show that the algorithm terminates after time exponential in the size of the input set $\mathcal{C}$, we first prove that the set of shortcuts increases in each iteration.

Lemma 26. *Let $i > 0$ be such that $\mathcal{R}_i$ was computed by Algorithm 3. Then $\mathcal{R}_{i-1} \subseteq \mathcal{R}_i$.*

Proof. We show the claim by induction on i. Since $\mathcal{R}_0 = \emptyset$, this is obviously true for $i = 1$. Consider now $i > 1$ and assume $\mathcal{R}_{i-2} \subseteq \mathcal{R}_{i-1}$. If $\mathcal{R}_{i-2} = \emptyset$, then it is easy to see that for every $Q \in \mathcal{P}$, $\texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-2}, Q) \subseteq \texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-1}, Q)$ holds. We now assume that $\mathcal{R}_{i-2}$ is nonempty.

Observe that for every $Q \in \mathcal{P}$ and every nonempty set $\mathcal{R}$ of shortcuts the equality

$$\texttt{possibilities}(\mathcal{C}, \mathcal{R}, Q) = \bigcup_{(P,\mathcal{X}) \in \mathcal{R}} \texttt{possibilities}(\mathcal{C}, \{(P, \mathcal{X})\}, Q)$$

holds. Thus, the set of possibilities increases when more shortcuts are available, i.e., $\texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-2}, Q) \subseteq \texttt{possibilities}(\mathcal{C}, \mathcal{R}_{i-1}, Q)$.

Let now $(P, \mathcal{X})$ be a shortcut in $\mathcal{R}_{i-1}$. This means that it had to be computed by $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_{i-2})$. Thus, there is a triple $(P, \mathcal{X}, \mathcal{X})$ in the final set $\mathcal{T}$ computed by this algorithm. The result $(P, \mathcal{X}, \mathcal{X})$ depends only on the possibilities chosen in the main loop of the algorithm. By the above observations, the same possibilities are still available in the run of $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_{i-1})$, and thus the triple $(P, \mathcal{X}, \mathcal{X})$ is also computed by $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_{i-1})$. This shows that the shortcut $(P, \mathcal{X})$ is also contained in $\mathcal{R}_i$. □

We now use this result to show that the algorithm uses at most exponential time.

Lemma 27. *Any execution of Algorithm 3 terminates after time at most exponential in the size of* $\mathcal{C}$*.*

Proof. By Lemma 26, the sets $\mathcal{R}_i$ always increase. Furthermore, there are only exponentially many possible shortcuts. Thus, even if the algorithm does not return `true`, it must return `false` after at most exponentially many computations of $\texttt{isTerminating}(\mathcal{C}, \mathcal{R}_i)$ and $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_i)$.

An execution of $\texttt{isTerminating}(\mathcal{C}, \mathcal{R}_i)$ takes only time polynomial in the size of $\mathcal{C}$. Indeed, the computation of bad states is saturated after at most $|\mathcal{P}|$ steps since $\mathcal{B}_0 \subseteq \mathcal{B}_1 \subseteq \cdots \subseteq \mathcal{P}$.

To analyze the runtime of $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_i)$, consider the number of times a triple of the set $\mathcal{T}$ is replaced by new triples. At each of these replacement steps, at most exponentially many successor triples are created since there are at most exponentially many elements in $\texttt{possibilities}(\mathcal{C}, \mathcal{R}_i, Q)$ for a predicate Q. Additionally, the set V_P in the third component of these triples will always be increased by one element, and thus can be replaced only $|\mathcal{P}|$ many times.

Thus, an execution of $\texttt{nextShortcuts}(\mathcal{C}, \mathcal{R}_i)$ can be seen as a set of trees rooted in the initial triples $(P, \{P\}, \emptyset)$, where each triple is connected to its successor triples and the leaves are labeled by triples of the form (P, R_P, R_P). The height of this tree is bounded by $|\mathcal{P}|$ and the branching degree is bounded exponentially in the size of $\mathcal{C}$. Since there are at most $|\mathcal{P}|$ many such trees, the total number of

nodes and thus the runtime of `nextShortcuts`$(\mathcal{C}, \mathcal{R}_i)$ is bounded exponentially in the size of $\mathcal{C}$.

Since Algorithm 3 makes exponentially many iteration steps, each of which takes exponential time, the overall runtime is still bounded exponentially in $\mathcal{C}$. □

This concludes the proof of the following result.

Theorem 28. *Termination of propagation nets of the form $\mathcal{N}_\mathcal{C}$ for normalized sets $\mathcal{C}$ of propagation rules can be decided in time exponential in the size of $\mathcal{C}$.*

In particular, one can decide the existence of finite Herbrand models for finite sets $\mathcal{C}$ of propagation rules. As described in the reductions of Sections 2 and 3.2, the above result yields an ExpTime-algorithm for this problem.

Corollary 29. *The existence of finite Herbrand models for finite sets of propagation rules can be decided in* ExpTime.

Deterministic Propagation Nets

The propagation net $\mathcal{N}_\mathcal{C}$ is called *deterministic* if all the clauses of $\mathcal{C}$ are *deterministic*, i.e., have at most one possibility. In this case, all places of $\mathcal{N}_\mathcal{C}$ have at most one outgoing arc.

Lemma 30. *Termination of deterministic propagation nets of the form $\mathcal{N}_\mathcal{C}$ for normalized sets $\mathcal{C}$ of propagation rules can be decided in time polynomial in the size of $\mathcal{C}$.*

Proof. It suffices to show that each of the sets $\mathcal{R}_i$ contains at most one (P, R_P) for each predicate $P \in \mathcal{P}$ and is computed by Algorithm 3 in polynomial time in the size of $\mathcal{C}$. By Lemma 26, the final set of shortcuts is then computed after at most polynomially many steps. Since each call to `isTerminating` takes only polynomial time, this bounds the overall runtime of the algorithm by a polynomial in the size of $\mathcal{C}$.

We show the claim by induction on i. The claim is trivially satisfied for $\mathcal{R}_0$. If the claim holds for $\mathcal{R}_{i-1}$, then `possibilities`$(\mathcal{C}, \mathcal{R}_{i-1}, P)$ contains at most one element which is computed in polynomial time. But then, in the computation of `nextShortcuts`$(\mathcal{C}, \mathcal{R}_{i-1})$, every triple is replaced by at most one successor triple. Since each such replacement step increases the second component of this triple, only polynomially many such steps are possible for each initial triple. This means that each initial triple $(P, \{P\}, \emptyset)$ gives rise to at most one final triple (P, R_P, R_P), and thus to at most one shortcut (P, R_P), in polynomial time. □

For every additional nondeterministic clause in the set $\mathcal{C}$, the runtime of the algorithm increases by an exponential factor due to the computation of all possibilities and all shortcuts in `possibilities`$(\mathcal{C}, \mathcal{R}, P)$ and `nextShortcuts`$(\mathcal{C}, \mathcal{R})$.

5 Hardness

To conclude the complexity analysis, we will present a reduction from linear language equations to finite sets of propagation rules. The equations are of the form

$$S_0 \cup S_1X_1 \cup \cdots \cup S_nX_n = T_0 \cup T_1X_1 \cup \cdots \cup T_nX_n$$

for finite sets $S_0, \ldots, S_n, T_0, \ldots, T_n$ of words over an alphabet Σ. A *solution* assigns finite sets of words over Σ to the variables X_i such that the equation holds. Deciding whether such an equation has a solution is an ExpTime-complete problem [1].

For easier presentation of our results, we will modify the presentation of the above linear language equations. First, we introduce a new variable A so that we can write T_0 as T_0A, which is of the same form as the other terms of the equation. We have to keep in mind that the value of A has to be restricted to $\{\varepsilon\}$.

Furthermore, we will transform the above equation over finite languages into a system of polynomially many *flat linear language inclusions* of the form

$$L_0X_0 \subseteq L_1X_1 \cup \cdots \cup L_nX_n$$

for $L_0, \ldots, L_n \subseteq \Sigma \cup \{\varepsilon\}$. By *flat* we mean that all coefficients contain only words of length at most 1.

To achieve this, we successively replace terms LX by $\bigcup_{r\in\Sigma} L_rX_r \cup L_\varepsilon X$ for new variables X_r, $L_r := \{w \mid wr \in L\}$, and $L_\varepsilon := \{w \mid w = \varepsilon \in L\}$ and introduce the new equations $X_r = \{r\}X$ for each $r \in \Sigma$.[6] This is a polynomial reduction, since the number of new variables is bounded by the sum of the lengths of the words occurring in the original equation. Once all equations are flat, we can easily replace them by equivalent flat inclusions.

Example 31. Consider the linear language equation

$$\{rs\} \cup \{s\}Y \cup X = \{r\}Y \cup \{s\}X \cup \{\varepsilon\}.$$

To flatten it, we introduce the new variable Z and arrive at the following equations. We use the variable A introduced above and abbreviate $\{r\}$ by r:

$$rZ \cup sY \cup X = rY \cup sX \cup A,\ Z = sA.$$

These equations are then split into the following 8 language inclusions:

$$\begin{aligned}\mathcal{I}_1 := \{&rZ \subseteq rY \cup sX \cup A,\ sY \subseteq rY \cup sX \cup A,\ X \subseteq rY \cup sX \cup A,\\ &rY \subseteq rZ \cup sY \cup X,\ sX \subseteq rZ \cup sY \cup X,\ A \subseteq rZ \cup sY \cup X,\\ &Z \subseteq sA,\ sA \subseteq Z\}.\end{aligned}$$

[6] Of course, if L_r is empty, then we do not need to introduce X_r.

In the following, we are concerned with solving finite sets $\mathcal{I}$ of inclusions of the above form. The set of variables occurring in $\mathcal{I}$ is denoted by $\mathrm{Var}(\mathcal{I})$. Note that this includes the special variable A which is restricted to be $\{\varepsilon\}$ in all solutions.

5.1 Translation to Clauses

We now translate any finite set $\mathcal{I}$ of language inclusions into a finite set $\mathcal{C}_\mathcal{I}$ of propagation rules that express the same restrictions as the inclusions. Recall that our inclusions are of the form

$$L_0X_0 \subseteq L_1X_1 \cup \cdots \cup L_nX_n$$

with $L_0, \ldots, L_n \subseteq N_R \cup \{\varepsilon\}$. We will treat each $r \in \Sigma$ as a unary function symbol, each variable $X \in \mathrm{Var}(\mathcal{I})$ as a unary predicate.

The intended meaning of the propagation rules we will construct is that a finite Herbrand model $\mathcal{H}$ of $\mathcal{C}_\mathcal{I}$ represents a solution θ of $\mathcal{I}$ with $\theta(X) = \{w \mid w(a) \in X^\mathcal{H}\}$. Thus, the clauses in $\mathcal{C}_\mathcal{I}$ have to impose the same restrictions on the variables as the inclusions in $\mathcal{I}$. Accordingly, we have to make sure that the special predicate A is always interpreted as $\{a\}$.

To express an inclusion $L_0X_0 \subseteq L_1X_1 \cup \cdots \cup L_nX_n$ by clauses, we use the following idea. The clauses have to restrict the interpretation of the variables such that every word $w \in \Sigma^*$ occurring on the left-hand side of the inclusion also occurs on the right-hand side. For each word w occurring in L_0X_0, we make a case analysis based on the first letter of w. We create one clause for the case $w = \varepsilon$, and one clause each for every possible first letter of w.

Example 32. Consider the inclusion $rZ \subseteq rY \cup sX \cup A$ from the set $\mathcal{I}_1$ constructed in Example 31. Every word w on the left-hand side of the inclusion has to begin with r, so the case analysis can be narrowed to one case. The corresponding clause is $Z(x) \to Y(x) \vee A(r(x))$, where x represents the suffix of w that starts after the initial r. Note that the term sX can never be responsible for this inclusion to be satisfied, which is why it is not represented in the clause.

The term $A(r(x))$ also deserves further explanation. Without knowledge about the special role of A, it is simply another variable. Thus, it is possible that the inclusion is satisfied by w being in the solution for A. To ensure that this does not happen, we introduce another clause $A(r(x)) \to \bot$. While the formal reduction will include terms like $A(r(x))$ on the right-hand side of clauses, in the examples we will leave them out for better readability.

Consider now another inclusion $X \subseteq rY \cup sX \cup A$, which has to be split according to s, r, and ε. For the case that a word w on the left-hand side begins with r, we introduce the clause $X(r(x)) \to Y(x) \vee A(r(x))$ as above. Similarly, for s we obtain $X(s(x)) \to X(x) \vee A(s(x))$. If $w = \varepsilon$, then w has to be in the solution for

A. This is expressed by the clause $X(a) \to A(a)$. Knowing that $A(a)$ always has to hold, we could also leave out this clause.

In order to formalize this translation, we need the following auxiliary mappings. For every language term αX with $\alpha \in \Sigma \cup \{\varepsilon\}$ and every $\beta \in \Sigma \cup \{\varepsilon\}$, we define the first order formula

$$(\alpha X)^\beta(x) := \begin{cases} X(x) & \text{if } \alpha = \beta \\ X(\beta(x)) & \text{if } \alpha = \varepsilon \text{ and } \beta \in \Sigma \\ \bot & \text{otherwise} \end{cases}.$$

For $L \subseteq \Sigma \cup \{\varepsilon\}$, we define $(LX)^\beta(x) := \bigvee_{\alpha \in L} (\alpha X)^\beta(x)$. We now define the set $\mathcal{C}_\mathcal{I} := \bigcup_{i=1}^4 \mathcal{C}_\mathcal{I}^i$ of propagation rules to contain the following clauses:

- Clauses ensuring that A is interpreted as the singleton set $\{a\}$:

  $\mathcal{C}_\mathcal{I}^1$: $\top \to A(a)$ and
  $\mathcal{C}_\mathcal{I}^2$: $A(r(x)) \to \bot$ for every $r \in \Sigma$.

- For every inclusion $L_0X_0 \subseteq L_1X_1 \cup \cdots \cup L_nX_n \in \mathcal{I}$ and every $\alpha \in L_0$, the clauses:

  $\mathcal{C}_\mathcal{I}^3$: $(\alpha X_0)^\varepsilon(a) \to (L_1X_1)^\varepsilon(a) \vee \cdots \vee (L_nX_n)^\varepsilon(a)$ and
  $\mathcal{C}_\mathcal{I}^4$: $(\alpha X_0)^r(x) \to (L_1X_1)^r(x) \vee \cdots \vee (L_nX_n)^r(x)$ for all $r \in \Sigma$.

Note that for $\alpha = s$, the left-hand side of the clauses in $\mathcal{C}_\mathcal{I}^3$ and $\mathcal{C}_\mathcal{I}^4$ with $r \neq s$ is $\bot$. These clauses can be removed from $\mathcal{C}_\mathcal{I}$ in a subsequent cleaning step.

Lemma 33. *$\mathcal{I}$ has a solution that maps A to $\{\varepsilon\}$ iff $\mathcal{C}_\mathcal{I}$ has a finite Herbrand model.*

Proof. Given a finite Herbrand model $\mathcal{H}$ of $\mathcal{C}_\mathcal{I}$, we define the mapping θ by $\theta(X) := \{w \in \Sigma^* \mid w(a) \in X^\mathcal{H}\}$. Note first that θ satisfies $\theta(A) = \{\varepsilon\}$, since $A^\mathcal{H} = \{a\}$ by $\mathcal{C}_\mathcal{I}^1$ and $\mathcal{C}_\mathcal{I}^2$. We now show that θ is a solution of $\mathcal{I}$.

Let $L_0X_0 \subseteq L_1X_1 \cup \cdots \cup L_nX_n$ be an inclusion in $\mathcal{I}$ and $w \in L_0\theta(X_0)$. This means that there is an $\alpha \in L_0$ such that $w = \alpha w'$ for a $w'(a) \in X_0^\mathcal{H}$. This implies that the atom $(\alpha X_0)^\alpha(w'(a)) = X_0(w'(a))$ is true in $\mathcal{H}$.

1. If $\alpha \in \Sigma$, then, since $\mathcal{H}$ satisfies the clauses in $\mathcal{C}_\mathcal{I}^4$, there must be $i \in \{1, \ldots, n\}$ and $\beta \in L_i$ such that $(\beta X_i)^\alpha(w'(a))$ is true in $\mathcal{H}$. Thus, $(\beta X_i)^\alpha(w'(a))$ cannot be $\bot$ and we must have one of the following cases:

   a) If $\beta = \alpha$, then $X_i(w'(a))$ is true in $\mathcal{H}$. Thus, we have $w' \in \theta(X_i)$, which implies that $w = \alpha w' = \beta w' \in L_i\theta(X_i)$, i.e., w also occurs on the right-hand side of the inclusion.

b) If $\beta = \varepsilon$, then $X_i(\alpha(w'(a)))$ is true in $\mathcal{H}$. Thus, we again have $w = \alpha w' \in \theta(X_i) \subseteq L_i\theta(X_i)$.

2. If $\alpha = \varepsilon$, then $w = w'$.

   a) If $w = \varepsilon$, then, since $\mathcal{H}$ satisfies the clauses in $\mathcal{C}^3_{\mathcal{I}}$, there must be $i \in \{1, \ldots, n\}$ and $\beta \in L_i$ such that $(\beta X_i)^\varepsilon(a)$ is true in $\mathcal{H}$. Thus, we must have $\beta = \varepsilon$ and $X_i(a)$ true in $\mathcal{H}$, which implies $w = \varepsilon \in \theta(X_i) \subseteq L_i\theta(X_i)$.

   b) If $w = sw''$ for some $s \in \Sigma$ and $w'' \in \Sigma^*$, then the atom $(\alpha X_0)^s(w''(a)) = X_0(s(w''(a))) = X_0(w(a))$ is true in $\mathcal{H}$. Since $\mathcal{H}$ satisfies the clauses in $\mathcal{C}^4_{\mathcal{I}}$, there is $i \in \{1, \ldots, n\}$ and $\beta \in L_i$ such that $(\beta X_i)^s(w''(a))$ is true in $\mathcal{H}$. We can now proceed as in the cases 1.a) and 1.b) to show that $w = sw'' \in L_i\theta(X_i)$.

On the other hand, if there is a solution θ of $\mathcal{I}$ with $\theta(A) = \{\varepsilon\}$, we can define a finite Herbrand model $\mathcal{H}$ of $\mathcal{C}_{\mathcal{I}}$ by $X^{\mathcal{H}} := \{w(a) \mid w \in \theta(X)\}$ for every variable $X \in \mathrm{Var}(\mathcal{I})$. Since $A^{\mathcal{H}} = \{a\}$, the clauses $\mathcal{C}^1_{\mathcal{I}}$ and $\mathcal{C}^2_{\mathcal{I}}$ are satisfied by $\mathcal{H}$. Let now $L_0X_0 \subseteq L_1X_1 \cup \cdots \cup L_nX_n \in \mathcal{I}$ be an inclusion of $\mathcal{I}$ and $\alpha \in L_0$. We will show that all propagation rules created from this inclusion are satisfied by $\mathcal{H}$.

Let $(\alpha X_0)^\varepsilon(a) \to (L_1X_1)^\varepsilon(a) \vee \cdots \vee (L_nX_n)^\varepsilon(a)$ be the corresponding propagation rule in $\mathcal{C}^3_{\mathcal{I}}$ and assume that the atom $(\alpha X_0)^\varepsilon(a)$ is true in $\mathcal{H}$. This can only be the case if $\alpha = \varepsilon$, and thus we have $a \in X_0^{\mathcal{H}}$ and $\varepsilon \in \theta(X_0) \subseteq L_0\theta(X_0)$. Since θ is a solution of $\mathcal{I}$, there is $i \in \{1, \ldots, n\}$ such that $\varepsilon \in L_i\theta(X_i)$. This implies that $\varepsilon \in L_i$ and $a \in X_i^{\mathcal{H}}$. Thus, the atom $(L_iX_i)^\varepsilon(a) = (\varepsilon X_i)^\varepsilon(a) = X_i(a)$ is true in $\mathcal{H}$, i.e., the clause is satisfied by $\mathcal{H}$.

We now consider the clause $(\alpha X_0)^r(x) \to (L_1X_1)^r(x) \vee \cdots \vee (L_nX_n)^r(x)$ of $\mathcal{C}^4_{\mathcal{I}}$ for some $r \in \Sigma$ and assume that $(\alpha X_0)^r(w(a))$ is true in $\mathcal{H}$. Thus, $(\alpha X_0)^r(w(a))$ is not $\bot$, which implies that $\alpha \in \{\varepsilon, r\}$.

a) If $\alpha = \varepsilon$, then $X_0(r(w(a)))$ is true in $\mathcal{H}$ and we have $rw(a) \in X_0^{\mathcal{H}}$, and thus $rw \in \theta(X_0) \subseteq L_0\theta(X_0)$.

b) If $\alpha = r$, then $w(a) \in X_0^{\mathcal{H}}$ and we again have $rw \in \{r\}\theta(X_0) \subseteq L_0\theta(X_0)$.

Since θ is a solution of $\mathcal{I}$, there is $i \in \{1, \ldots, n\}$ with $rw \in L_i\theta(X_i)$, which implies that there is $\beta \in L_i$ such that $rw = \beta w'$ for a word $w' \in \theta(X_i$.

a) If $\beta = \varepsilon$, then $rw(a) = w'(a) \in X_i^{\mathcal{H}}$, and thus $(\beta X_i)^r(w(a)) = X_i(r(w(a)))$ is true in $\mathcal{H}$.

b) If $\beta = r$, then $w(a) = w'(a) \in X_i^{\mathcal{H}}$, and again $(\beta X_i)^r(w(a)) = X_i(w(a))$ is true in $\mathcal{H}$.

In both cases, $\mathcal{H}$ satisfies $(L_iX_i)^r(w(a))$, and thus the whole clause. □

It is also important to note that the size of $\mathcal{C}_\mathcal{I}$ is polynomial in the size of $\mathcal{I}$: For each inclusion, $|\Sigma| + 1$ clauses are created.

Example 34. Consider again the set $\mathcal{I}_1$ from Example 31. The following is the corresponding set of clauses:

$$\begin{aligned}\mathcal{C}_1 := \{&\top \to A(a),\ A(r(x)) \to \bot,\ A(s(x)) \to \bot,\\ &Z(a) \to \bot,\ Z(r(x)) \to \bot,\ Z(s(x)) \to A(x),\ A(x) \to Z(s(x)),\\ &A(a) \to X(a),\ X(a) \to A(a),\\ &Y(x) \to Z(x) \vee X(r(x)),\ Z(x) \to Y(x),\ X(r(x)) \to Y(x)\\ &X(x) \to Y(x) \vee X(s(x)),\ Y(x) \to X(x),\ X(s(x)) \to X(x)\}\end{aligned}$$

These are the same propagation rules as in Example 4, up to a renaming of the predicates.

This reduction allows us to conclude the following.

Theorem 35. *Deciding the existence of finite Herbrand models for finite sets of propagation rules is* ExpTime*-hard.*

6 Summary

From Corollary 29 and Theorem 35, we know that the following holds.

Theorem 36. *Deciding the existence of finite Herbrand models for finite sets of propagation rules is an* ExpTime*-complete problem.*

Viewed from a different perspective, Algorithm 3 and the reduction in Lemma 33 yield a new ExpTime-algorithm for deciding solvability of a linear language equation of the form presented in the previous section. While the original decision procedure from [1] constructs a tree automaton of size exponential in the cumulative size of the coefficients L occurring in the equation and uses the linear-time emptiness test for this automaton, our algorithm constructs a polynomial-size propagation net and uses an algorithm that is worst-case exponential, but exhibits a better behavior if the structure of the equation is "easy".

Here, "easy" means that the constructed set $\mathcal{C}_\mathcal{I}$ of propagation rules contains few nondeterministic clauses. As described in Section 4, for deterministic problems we can decide solvability in polynomial time, and every nondeterministic clause increases the runtime of the algorithm by an exponential factor due to the computation of all possibilities and all shortcuts in $\texttt{possibilities}(\mathcal{C}_\mathcal{I}, \mathcal{R}, Q)$ and $\texttt{nextShortcuts}(\mathcal{C}_\mathcal{I}, \mathcal{R})$.

We now analyze what kind of solutions are produced by our algorithm. If we consider the *size* of a solution to be the maximal length of the words occurring in

it, then the algorithm produces solutions of minimal size. This is due to the fact that the algorithm stops as soon as it is possible to construct a terminating firing sequence from the current shortcuts. The shortcuts computed in the i-th iteration represent all replacement sequences of height $\leq i$ (see Lemmata 21 and 23).

Assume to the contrary that the algorithm terminated in the i-th iteration and that there is a terminating firing sequence of height $j < i$. By Lemma 25, the existence of this firing sequence implies that $\texttt{isTerminating}(\mathcal{C}, \mathcal{R}_j)$ already returns `true`, which contradicts the assumption. The claim now follows from the fact that the height of firing sequences and the size of the corresponding solutions are the same (see the proofs of Lemmata 33, 3, and 10).

7 Conclusions

We have shown that the existence of finite Herbrand models for finite sets of propagation rules can be decided in ExpTime. The problem is even ExpTime-hard since solving linear language equations is ExpTime-hard. This also yields a new algorithm solving these linear language equations. This new approach has the advantage that it exploits the structure of the equations to guide the search for a solution.

In future work, we want to analyze the usefulness of the solutions of minimal size that are computed by the algorithm. Optionally, we may modify the algorithm to output minimal solutions w.r.t. a different order. We also want to implement the algorithm and compare it with an implementation of the naive tree automaton construction. To this end, we will have to design optimizations to our algorithm.

Another interesting open question is whether the presented approach can also be applied to finite sets of arbitrary clauses with unary predicates and function symbols. The formalism of propagation nets is certainly powerful enough to reflect this change, but the decision procedure would also have to be adapted.

Acknowledgements

We would like to thank Prof. Franz Baader for his helpful comments on early drafts of this report.

References

[1] Franz Baader and Paliath Narendran. Unification of concept terms in description logics. *Journal of Symbolic Computation*, 31(3):277–305, 2001.

[2] Peter Baumgartner, Alexander Fuchs, Hans de Nivelle, and Cesare Tinelli. Computing finite models by reduction to function-free clause logic. *Journal of Applied Logic*, 7(1):58–74, 2009.

[3] Jean-Camille Birget. State-complexity of finite-state devices, state compressibility and incompressibility. *Mathematical Systems Theory*, 26(3):237–269, 1993.

[4] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[5] Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007.

[6] B. Dreben and W. D. Goldfarb. *The Decision Problem: Solvable Classes of Quantificational Formulas*. Addison-Wesley, 1979.

[7] William H. Joyner Jr. Resolution strategies as decision procedures. *Journal of the ACM*, 23(3):398–417, 1976.

[8] Richard E. Ladner, Richard J. Lipton, and Larry J. Stockmeyer. Alternating pushdown and stack automata. *SIAM Journal on Computing*, 13(1):135–155, 1984.

[9] Alexander Leitsch. *The Resolution Calculus*. Springer-Verlag, 1997.

[10] Nicolas Peltier. Model building with ordered resolution: Extracting models from saturated clause sets. *Journal of Symbolic Computation*, 36(1-2):5–48, 2003.

[11] Carl A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, Universität Bonn, 1962.

[12] Wolfgang Reisig. *Petri Nets: An Introduction*. Springer-Verlag, 1985.

[13] Giora Slutzki. Alternating tree automata. *Theoretical Computer Science*, 41:305–318, 1985.

[14] Moshe Y. Vardi and Pierre Wolper. Automata theoretic techniques for modal logics of programs (extended abstract). In *Proc. of the 16th Annual ACM Symp. on Theory of Computing (STOC'84)*, pages 446–456. ACM, 1984.

[15] Jian Zhang. Constructing finite algebras with FALCON. *Journal of Automated Reasoning*, 17:1–22, 1996.