UPPSALA THESES IN COMPUTING SCIENCE
29

On Simultaneous Rigid E-Unification

Margus Veanes

Computing Science Department
Uppsala University

Thesis for the Degree of
Doctor of Philosophy

UPPSALA 1997

On Simultaneous Rigid E-Unification

Margus Veanes

A Dissertation submitted in partial fulfilment of the requirements for the Degree of Doctor of Philosophy at Computing Science Department, Uppsala University.

Uppsala University
Computing Science Department
Box 311, S-751 05 Uppsala, Sweden

Uppsala Theses in Computing Science 29
ISSN 0283-359X
ISBN 91-506-1217-4

(Dissertation for the Degree of Doctor of Philosophy in Computing Science presented at Uppsala University in 1997)

Abstract

Veanes, M. 1997: On Simultaneous Rigid E-Unification, *Uppsala Theses in Computing Science 29*. 122pp. Uppsala. ISSN 0283-359X, ISBN 91-506-1217-4.

Automated theorem proving methods in classical logic with equality that are based on the Herbrand theorem, reduce to a problem called Simultaneous Rigid E-Unification, or SREU for short. Recent developments show that SREU has also close connections with intuitionistic logic with equality, second-order unification, some combinatorial problems and finite tree automata.

We show new decidability results of various cases of SREU. In particular, we improve upon the undecidability result of SREU by showing its undecidability in a very restricted case, in fact the minimal known case. We prove the decidability of some cases of SREU under certain restrictions regarding the number of variables and other syntactical criteria. To prove the (computational) complexity of some cases, we reduce them to certain decision problems of finite tree automata. The complexity of the latter problems is studied first. In connection with that, we present a survey of closely related problems and give a comparison with the corresponding results in classical automata theory.

These results are applied in the context of intuitionistic logic with equality, to obtain a complete classification of its prenex fragment in terms of the quantifier prefix: the $\exists\exists$-fragment is shown undecidable and the $\forall^*\exists\forall^*$-fragment is shown decidable. These results have further implications for proof search in intuitionistic logic with equality.

We also improve upon a number of other recent undecidability results that are related to the so-called Herbrand Skeleton problem and are of fundamental importance in automated theorem proving in classical logic with equality. In that context we also prove, as our main tool, a logical theorem that we believe is of independent interest.

Margus Veanes, Computing Science Department, Uppsala University, Box 311, S-751 05 Uppsala, Sweden.

ISSN 0283-359X
ISBN 91-506-1217-4

With love, to my wife Katrine
and sons Margus and Jaan

Acknowledgments

This thesis would not exist if it wasn't for my supervisor *Dr Andrei Voronkov*, who introduced me into this subject for a year and a half ago; his expertise in the area and insightful comments throughout the course of this work have been of utmost importance.

The person whose cooperation has meant a lot for the development of this thesis is *Professor Yuri Gurevich* from the University of Michigan. His visit in Uppsala in the summer of 1996 led to many important breakthroughs. I will always remember the scientific discussions we had, behind a cup of cappucino, in the "Dungeon" in the Old Town of Stockholm.

I want to thank my colleague *Dr Jonas Barklund* for his support in all situations and many valuable discussions.

The courses given by the members of the mathematical logic group of Uppsala University have had a great influence on my work. In particular, I want to thank *Professor Viggo Stoltenberg-Hansen* and *Dr Erik Palmgren* for contributing to the development of my mathematical skills.

The numerous discussions with *Dr Anatoli Degtyarev* and *Dr Evgeny Dantsin* have been most enlightening. Many thanks goes also to my colleague and friend *Pierangelo Dell'Acqua* for always being concerned about me. I am also grateful to all the other persons who have helped me during these years.

My wife *Katrine Veanes* and our two sons *Margus* (3.4 years) and *Jaan* (1.8 years) mean everything to me. Without their support and encouragement, none of this would have been possible. I also thank my mother for raising me to believe in the importance of education.

Contents

Introduction

The central theme of this thesis is *simultaneous rigid E-unification*. Originally, simultaneous rigid E-unification was introduced in the context of automated theorem proving in classical logic with equality, but has thereafter been shown to have important connections to other research areas, such as intuitionistic logic with equality, second-order unification, some combinatorial problems and finite tree automata. In the thesis we look closely at some of these connections. In particular, we investigate

- the decidability of various fragments of simultaneous rigid E-unification,
- the decidability of some fundamental decision problems of automated theorem proving related to the Herbrand theorem,
- the complexity of basic decision problems of finite tree automata and
- the decidability of some fragments of intuitionistic logic with equality.

We show new decidability and complexity results in all those areas and point out some open problems.

This chapter is organized as follows. We start by giving a brief overview of the so-called rigid variable methods that arise in automated theorem proving methods in classical logic based on the Herbrand theorem. We then give a brief history of simultaneous rigid E-unification and illustrate how it arises in the rigid variable methods when equality is allowed. We explain also the relationships between the notions: (simultaneous) *unification*, *E-unification*, *rigid E-unification* and *simultaneous rigid E-unification*. Finally, we give an overview of the rest of the thesis.

1.1 RIGID VARIABLE METHODS

Automated theorem proving in classical first-order logic deals with the *validity problem* of closed first-order formulas:

Given a closed formula φ, is φ valid?

There is a category of automated theorem proving methods, known as the *tableau* methods and the *matrix* methods, the theorectical foundation of which is provided by the *Herbrand theorem* [74]. Below we illustrate both methods briefly. Collectively, such methods are referred to as the *rigid variable methods* [157].

We can consider, without loss of generality, first-order languages with at least one constant. So the set of *ground*[1] terms is always nonempty. One popular form of the Herbrand theorem is this:

> *A closed formula $\exists\vec{x}\varphi(\vec{x})$, where φ is quantifier free, is valid if and only if there exists a positive integer m and a sequence of ground terms $\vec{t}_1, \ldots, \vec{t}_m$ such that the disjunction $\varphi(\vec{t}_1) \vee \cdots \vee \varphi(\vec{t}_m)$ is valid.*

Informally, we refer to the number m as the *multiplicity*. The reason why it is sufficient to consider formulas in this dual *Skolemized prenex* form, is that any closed formula can easily be transformed into this form that is valid if and only if the original formula is valid.

The tableau and the matrix methods are all *refutation* systems[2], but one can easily dualize these methods to get a direct proof instead of a refutation. If we ignore the details in such dualized versions, we can identify the following procedure underlying them. We call it the *principal procedure* of rigid variable methods. Let $\exists\vec{x}\varphi(\vec{x})$ be a closed formula that we wish to decide the validity of, where $\varphi(\vec{x})$ is quantifier free. Proceed as follows:

Step I Choose a multiplicity m.

Step II Check if there exists a sequence of ground terms $\vec{t}_1, \ldots, \vec{t}_m$ such that the disjunction $\varphi(\vec{t}_1) \vee \cdots \vee \varphi(\vec{t}_m)$ is valid.

Step III If such a sequence of terms exists then $\exists\vec{x}\varphi(\vec{x})$ is valid, otherwise increase m and return to Step II.

Let us apply this procedure to the valid formula

$$\exists x\,(P(0) \vee P(1) \Rightarrow P(x)). \tag{1.1}$$

Choose $m = 1$ at Step I. It is easy to see that Step II fails, so increase m by 1 at Step III and re-execute Step II. Now, Step II succeeds with a solution given by the terms 0 and 1.

[1] We follow the custom of calling variable free terms *ground*, whereas formulas without free variables are called *closed*. A closed formula without quantifiers is also called ground.

[2] Instead of a direct proof, the negation of the formula is shown unsatisfiable.

There is of course no general way to compute an upper bound for the multiplicity directly from the formula, because the validity problem of classical logic (even without equality) is undecidable. The Herbrand theorem guarantees, however, that if the formula is valid, then the above procedure terminates eventually. Once the multiplicity is fixed, Step II reduces to (*simultaneous*) *unification* if we consider logic without equality. If equality is allowed, then Step II reduces to *simultaneous rigid E-unification*. Let us take a brief look at the tableau and the matrix methods in logic without equality.

Tableau Methods

A proof of a closed formula φ using a tableau method amounts to a refutation of $\neg\varphi$. The formula $\neg\varphi$ is expanded into a tree or *tableau* by using tableau expansion rules. Each branch of such a tree should be thought of as a conjunction of formulas appearing in it and the tree itself as a disjunction of its branches. A refutation of $\neg\varphi$ has been obtained if each branch is *inconsistent*, i.e., contains an atom and the negation of the same atom.

It is necessary to consider the so-called *free variable* versions of the tableau methods in order to avoid the huge level of non-determinism that arises when choosing the terms in the universal quantifier tableau expansion rules [36, 40, 47]. In the free variable based tableau methods, a tableau proof amounts to finding a *substitution*[3] that replaces all the variables in the tableau with ground terms such that all branches become inconsistent, which is tantamount to *simultaneous unification*. Figure 1.1 illustrates a tableau proof of Formula (1.1). For simplicity, the negation of Formula (1.1) has been slightly transformed. So all the branches of the tableau in Figure 1.1 can be made inconsistent if the following system of equations has a unifier:

$$\{x_0 \approx 0, \quad x_1 \approx 1\}.$$

Matrix Methods

The *matrix characterization* of validity [119] involves a two-dimensional view of formulas with disjunctions being displayed horizontally and conjunctions being displayed vertically. For example, the *mating method* [1] is based on this characterization. Let us consider a proof of Formula (1.1) by using the mating method. First, the negation of Formula (1.1) is transformed into the following equivalent one (in negation normal form and with the scope of the quantifiers being minimized):

$$(P(0) \vee P(1)) \wedge \forall x \neg P(x).$$

[3] A mapping from variables to terms.

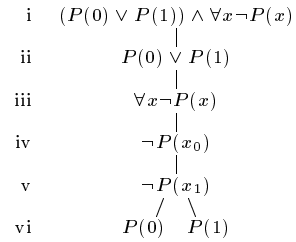


Figure 1.1: A tableau of $(P(0) \vee P(1)) \wedge \forall x \neg P(x)$ with the free variable tableau system of Fitting [47]. Lines ii and iii result from line i. Lines iv and v result from line iii by *two* applications of the γ-rule. Line vi results from line ii. Both branches of the tableau can be made inconsistent with a substitution that maps x_0 to 0 and x_1 to 1.

Next, the subformula $\forall x \neg P(x)$ is duplicated and the quantifiers are removed from the resulting formula. So, a formula of the following form is obtained:

$$(P(0) \vee P(1)) \wedge \neg P(x_0) \wedge \neg P(x_1).$$

Now, the matrix of this formula looks like this:

$$\begin{bmatrix} P(0) & P(1) \\ \neg P(x_0) & \\ \neg P(x_1) & \end{bmatrix}$$

This matrix has two "vertical paths":

$$\{P(0), \neg P(x_0), \neg P(x_1)\} \text{ and } \{P(1), \neg P(x_0), \neg P(x_1)\}.$$

Both paths can be simultaneously made inconsistent (to contain an atom and its negation), through an application of a substitution that maps x_0 to 0 and x_1 to 1. This proves that the negation of Formula (1.1) is unsatisfiable.

Remarks We can note that the necessity to increase the multiplicity at Step III, while searching for a proof of Formula (1.1) with the principal procedure, corresponds precisely to the two necessary applications of the γ-rule in Figure 1.1 or to the duplication of the subformula $\forall x \neg P(x)$ in the mating proof of Formula (1.1).

Background

Automated theorem proving methods in classical logic can be divided roughly into two categories. The first category comprises methods that are refinements of Robinson's *resolution principle* [124] that descends from techniques developed already by Herbrand [74]. The first general method to handle equality in resolution based methods is based on *paramodulation* [123].

Although there have been other approaches, the main line of research in resolution theorem proving with equality has been dominated by various improvements of paramodulation [40].

The other category of methods, that we considered above, is based on *semantic tableau* or *sequent calculus* based proof systems, originally developed by Beth [9] and further studied by Smullyan [137]. Independently, similar methods were introduced by Kanger [83]. We refer to such methods under the name of *tableau*. Tableau methods are to a large extent based on Gentzen's work [61]. The *matrix characterization* of provability (where free variables were used for the first time) was introduced by Prawitz [118, 119] for formulas in conjunctive normal form and was later generalized to arbitrary formulas, independently, by Andrews [1] and Bibel [10]. Related ideas appear already in Quine [120]. The study of equality reasoning in sequent calculus based methods was already started by Wang [158]. Important pioneering work in this connection was done by Kanger [84].

Both categories are addressed and compared by several authors. For a general comparison and an introduction to tableau and resolution systems for arbitrary formulas, see Fitting [47]. The close correspondence between tableau and sequent calculus systems is described in Smullyan [137]. For a comprehensive treatment of equality reasoning in automated theorem proving in general see the tutorial by Degtyarev and Voronkov [40]. Futher comparisons can be found, e.g., in Eder [45, 46], Bibel and Eder [11] and Ophelders and de Swart [108].

1.2 SIMULTANEOUS RIGID E-UNIFICATION

Simultaneous Rigid E-Unification was introduced by Gallier, Raatz and Snyder [57], who showed that the method of matings by Andrews [1] can be extended to logic with equality by incorporating simultaneous rigid E-unification. Considering a free variable tableau method, the key observation is the following. The problem of finding a substitution that makes a branch inconsistent amounts to solving the following problem, called *rigid E-unification*:

> Given a finite set of equations E and an equation e, does there exist a substitution θ such that $E\theta$ and $e\theta$ are ground[4] and $e\theta$ is a logical consequence of $E\theta$? Such a substitution is said to *solve* the *rigid equation* $E \models_{\forall} e$.

Now, the problem of finding a substitution θ that simultaneously makes *all* the branches inconsistent, corresponds to solving a *system* of such rigid

[4]The expression $X\theta$, where X is E or e, denotes the result of replacing each free variable x in X with the term $\theta(x)$. The groundness condition of the result is for technical reasons only, it is not part of the standard definition.

equations. This problem is called *simultaneous rigid E-unification* or SREU for short. There are several papers [50, 52, 53, 56, 58] that explain in detail how SREU arises in the mating method.

SREU versus E-Unification and Unification

The first decidability and NP-completeness proof of *rigid E-unification* was given by Gallier, Narendran, Plaisted and Snyder [52, 56]. Since then, the decidability (and NP-completeness) of rigid E-unification has been reestablished by other authors, e.g., Plaisted [116], de Kogel [27, 28] and Choi [19].

In contrast, the problem of *E-unification* is undecidable. A good example is the undecidability of weak equality in Combinatory Logic due to Scott and Curry (cf Hindley and Seldin [75, Chapter 5]). Let $\cdot$ be a binary function symbol and let $\mathbf{S}$ and $\mathbf{K}$ be two constants. The undecidability of weak equality implies that the following problem is undecidable:

Given ground terms t and s, is it the case that[5]

$$\left.\begin{array}{r} \forall x\,\forall y\,(\mathbf{K}\cdot x)\cdot y \approx x \\ \forall x\,\forall y\,\forall z\,((\mathbf{S}\cdot x)\cdot y)\cdot z \approx (x\cdot z)\cdot(y\cdot z) \end{array}\right\} \models s \approx t \quad ?$$

SREU was proved undecidable by Degtyarev and Voronkov [34]. Before that result, there were several faulty statements about the decidability of SREU, e.g., that SREU is NP-complete [50, 52, 56], EXPTIME-complete [66] and even NEXPTIME-complete [65]. The undecidability of SREU was quite unexpected and implied the undecidability of several other fundamental decision problems in automated theorem proving [30].

Simultaneous unification reduces to unification. Unification can be solved in almost linear time [97] and even in linear time [114] if more complex data structures are used. It is also known that unification is P-complete [44, 161].

An Example

Let us see how SREU can be used in a rigid variable method, but instead of using any particular method let us consider the principal procedure. First of all, it is easy to show that there is a simple reduction from Step II to SREU without introducing any new function symbols.[6] Assume that we want to decide the validity of the following formula:

$$\exists x((c \approx f(c) \Rightarrow x \approx g(c)) \wedge (c \approx g(c) \Rightarrow x \approx f(c))). \tag{1.2}$$

[5] We use '$\approx$' for the formal equality sign.

[6] Such a reduction is given for example in Voda and Komara [151].

Choose the multiplicity m to be 1 at Step I. The problem at Step II is now to decide if there exists a substitution that makes the following formula ground and valid:

$$(c \approx f(c) \Rightarrow x \approx g(c)) \wedge (c \approx g(c) \Rightarrow x \approx f(c)).$$

Such a substitution exists if and only if this system of two rigid equations is solvable:

$$\begin{aligned} \{\, c \approx f(c) \,\} \quad &\models_\forall \quad x \approx g(c), \\ \{\, c \approx g(c) \,\} \quad &\models_\forall \quad x \approx f(c). \end{aligned}$$

This system is not solvable, because if a substitution θ solves the first rigid equation then $x\theta = g(f^k(c))$ for some $k \geq 0$, and if θ solves the second rigid equation then $x\theta = f(g^k(c))$ for some $k \geq 0$. So let us increase m by one and return to Step II. The problem at Step II is now to decide if there exists a substitution that makes the following formula ground and valid:

$$\begin{aligned} &[(c \approx f(c) \Rightarrow x \approx g(c)) \wedge (c \approx g(c) \Rightarrow x \approx f(c))] \vee \\ &[(c \approx f(c) \Rightarrow y \approx g(c)) \wedge (c \approx g(c) \Rightarrow y \approx f(c))]. \end{aligned}$$

So let us see how this problem can be solved by using SREU. The easiest way to see this is to first transform the formula into *conjunctive normal form* (a conjunction of disjunctions of literals):

$$\begin{aligned} & (c \not\approx f(c) \vee x \approx g(c) \vee c \not\approx f(c) \vee y \approx g(c)) \\ \wedge \quad & (c \not\approx f(c) \vee x \approx g(c) \vee c \not\approx g(c) \vee y \approx f(c)) \\ \wedge \quad & (c \not\approx g(c) \vee x \approx f(c) \vee c \not\approx f(c) \vee y \approx g(c)) \\ \wedge \quad & (c \not\approx g(c) \vee x \approx f(c) \vee c \not\approx g(c) \vee y \approx f(c)). \end{aligned}$$

Now, for each conjunct construct a rigid equation $E \models_\forall e$, where E is the set of all the equations in it that occur negatively and e is one of the positive equations (chosen nondeterministically). By applying this operation to all the conjuncts of the above formula, we can get a system of four rigid equations:

$$\begin{aligned} \{\, c \approx f(c) \,\} \quad &\models_\forall \quad y \approx g(c), \\ \{\, c \approx f(c),\ c \approx g(c) \,\} \quad &\models_\forall \quad x \approx g(c), \\ \{\, c \approx g(c),\ c \approx f(c) \,\} \quad &\models_\forall \quad x \approx f(c), \\ \{\, c \approx g(c) \,\} \quad &\models_\forall \quad x \approx f(c). \end{aligned}$$

This system is solvable, e.g., with a substitution θ such that $x\theta = f(g(c))$ and $y\theta = g(f(c))$. We conclude that Formula 1.2 is valid according to Step III.

1.3 OUTLINE OF THE THESIS

Chapter 2 We introduce the main notions and definitions that are used throughout the thesis. Some notions are used only locally within one chapter, in those cases the definitions are given first in the preliminaries of the corresponding chapter.

Chapter 3 We present a new proof of the undecidability of SREU. This proof implies that already a very small, actually smallest known, fragment of SREU is undecidable. Due to the elementary nature of the proof, it brings out and clarifies the properties of SREU that make it an undecidable problem. We present a short survey of the earlier undecidability proofs.

Chapter 4 The Herbrand theorem plays a fundamental role in automated theorem proving and the so-called *Herbrand Skeleton problem* has been of considerable interest. In the general case it is effectively equivalent to SREU and thus undecidable. In this chapter we improve upon a number of undecidability results related to the Herbrand Skeleton problem. The main result is a logical theorem, that we call the *Partisan Corroboration Theorem*, that we believe is of independent interest, and is our main tool in proving the undecidability results.

Chapter 5 Finite tree automata are generalizations of classical finite automata to automata that accept trees of symbols, not just strings of symbols. They are a fundamental tool in various areas of computer science. We focus on the following basic decision problems of finite tree automata: *non-emptiness* and *intersection non-emptiness*. The first problem is shown P-complete and the second one EXPTIME-complete. We include a short survey of closely related problems and draw some general conclusions from this.

Chapter 6 We show that SREU with one variable is decidable. Moreover, we show that this problem is EXPTIME-complete. However, if the number of rigid equations is bounded by a constant then the problem is P-complete. So, the intractability of SREU with one variable is strongly related to the *number* of rigid equations and not their *size*. We also show the decidability of SREU when one allows several variables, but each rigid equation either contains one variable, or has a ground left-hand side and an equality between two variables on the right-hand side.

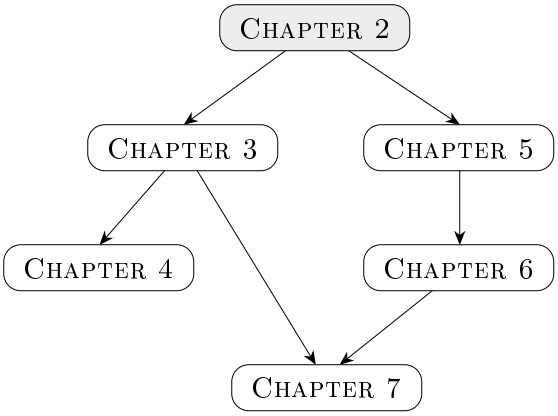


Figure 1.2: Dependencies between the main results of the chapters.

Chapter 7 The prenex fragment of intuitionistic logic is the collection of all intuitionistically provable prenex formulas. We give a complete classification of decidability of the prenex fragment of intuitionistic logic with equality, in terms of the quantifier prefix: the $\exists\exists$-fragment is shown undecidable and the $\forall^*\exists\forall^*$-fragment is shown decidable. At the the end of this chapter we compare these results with the corresponding results in classical logic.

Chapter 8 We state the main contributions of the thesis and give a list of all the results that are known about SREU. Finally, we point out some open problems and discuss future work.

1.4 HOW TO READ THE THESIS

Chapters 3–7 are self-contained and can be read independently, consulting Chapter 2 only when necessary. The dependencies between the main results in the thesis are illustrated with the the diagram in Figure 1.2.

1.5 SOURCE MATERIAL

This thesis is mainly based on the following material.

- M. Veanes. Uniform representation of recursively enumerable sets with simultaneous rigid E-unification. UPMAIL Technical Report 126, Uppsala University, Computing Science Department, July 1996.

- M. Veanes. The undecidability of simultaneous rigid E-unification with two variables. To appear in *Proceedings of Kurt Gödel Colloquium KGC'97*, 1997.
- M. Veanes. On computational complexity of basic decision problems of finite tree automata. UPMAIL Technical Report 133, Uppsala University, Computing Science Department, January 1997, Submitted.
- Y. Gurevich and M. Veanes. Some undecidable problems related to the Herbrand theorem. UPMAIL Technical Report 138, Uppsala University, Computing Science Department, March 1997, Submitted.
- A. Degtyarev, Y. Gurevich, P. Narendran, M. Veanes, and A. Voronkov. The decidability of simultaneous rigid E-unification with one variable. UPMAIL Technical Report 139, Uppsala University, Computing Science Department, March 1997, Submitted.

The order of the authors is purely alphabetical and is not intended to indicate the extent of the individual contributions. The main ideas behind some of the results that are presented in the thesis have been obtained in collaboration with some of the coauthors. However, all the written proofs and the presentation of the material in this thesis is the result of the individual effort of the author of the thesis.

PRELIMINARIES

2.1 FIRST-ORDER LOGIC

We follow Chang and Keisler [18] regarding first-order languages and structures. We always assume, unless otherwise stated, that the first-order languages that we are dealing with are languages with equality and contain only function symbols. A *signature* is a collection of function symbols with fixed arities. A function symbol of arity 0 is called a *constant*. We use Σ or Γ, possibly with an index, to denote a signature. In general, a signature is assumed to contain at least one constant.

Terms and Formulas

Terms and formulas are defined in the standard manner. We refer to terms and formulas collectively as *expressions*. In the following let X be an expression or a set of expressions.

We write $\Sigma(X)$ for the *signature of* X, i.e., the set of all function symbols that occur in X and $\mathcal{V}(X)$ for the set of all free variables in X. Let $\vec{x} = x_1, x_2, \ldots, x_n$ be a sequence of distinct variables such that $\mathcal{V}(X) \subseteq \{x_1, \ldots, x_n\}$. We use the metanotation $X(\vec{x})$ to indicate that $\mathcal{V}(X) \subseteq \{x_1, x_2, \ldots, x_n\}$. Let $\vec{t} = t_1, t_2, \ldots, t_n$ be a sequence of terms, then $X(\vec{t})$ denotes the result of replacing each (free) occurrence of x_i in X by t_i for $1 \leq i \leq n$. By a *substitution* we mean a function from variables to terms. We use θ, possibly with and index, to denote a substitution. We write $X\theta$ for $X(\theta(x_1), \theta(x_2), \ldots, \theta(x_n))$.

We say that X is *closed* or *ground* if $\mathcal{V}(X) = \emptyset$. By $\mathcal{T}_\Sigma$ or simply $\mathcal{T}$ we denote the set of all ground terms over the signature Σ. A substitution is called *ground* if its range consists of ground terms. A closed formula is called a *sentence*. Since there are no relation symbols all the atomic formulas are *equations*, i.e., of the form $t \approx s$ where t and s are terms and '$\approx$' is the formal equality sign.

First-Order Structures

First-order structures are (in general) denoted by upper case gothic letters like $\mathfrak{A}$ and $\mathfrak{B}$. A first-order structure in a signature Σ is called a Σ-*structure*. For $f \in \Sigma$ we write $f^{\mathfrak{A}}$ for the interpretation of f in $\mathfrak{A}$.

For a sentence or a set of sentences X, $\mathfrak{A} \models X$ means that the structure $\mathfrak{A}$ is a *model of* or *satisfies* X according to Tarski's truth definition. A set of sentences is called *satisfiable* if it has a model. If X and Y are (sets of) sentences then $X \models Y$ means that Y is a *logical consequence* of X, i.e., that every model of X is a model of Y. We write $\models X$ to say that X is *valid*, i.e., true in all models.

By the *free algebra over* Σ we mean the Σ-structure $\mathfrak{A}$, with universe $\mathcal{T}_\Sigma$, such that for each n-ary function symbol $f \in \Sigma$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$, $f^{\mathfrak{A}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$. We let $\mathcal{T}_\Sigma$ also stand for the free algebra over Σ.

Let E be a set of ground equations. Define the equivalence relation $=_E$ on $\mathcal{T}$ by $s =_E t$ iff $E \models s \approx t$. By $\mathcal{T}_{\Sigma/E}$ (or simply $\mathcal{T}_{/E}$) we denote the quotient of $\mathcal{T}_\Sigma$ over $=_E$. Thus, for all $s, t \in \mathcal{T}$,

$$\mathcal{T}_{/E} \models s \approx t \quad \Leftrightarrow \quad E \models s \approx t.$$

We call $\mathcal{T}_{/E}$ the *canonical model* of E.

2.2 SIMULTANEOUS RIGID E-UNIFICATION

A *rigid equation* is an expression of the form $E \models_\forall s \approx t$ where E is a finite set of equations, called the *left-hand side* of the rigid equation, and s and t are arbitrary terms; the equation $s \approx t$ is called the *right hand side* of the rigid equation. A *system* of rigid equations is a finite set of rigid equations. A substitution θ is a *solution of* or *solves* a rigid equation $E \models_\forall s \approx t$ if

$$\models \quad (\bigwedge_{e \in E} e\theta) \Rightarrow s\theta \approx t\theta,$$

and θ is a *solution of* or *solves* a system of rigid equations if it solves each member of that system. The problem of solvability of systems of rigid equations is called *simultaneous rigid E-unification* or SREU for short. Solvability of a single rigid equation is called *rigid E-unification*.

2.3 TERM REWRITING

In some cases it is convenient to use ground term rewriting techniques at metalevel [42, 78] when reasoning about equations. Let $\longrightarrow$ be a binary relation on terms. We define first some well-known properties of $\longrightarrow$. The

reflexive and transitive closure of $\longrightarrow$ is denoted by $\stackrel{*}{\longrightarrow}$. The relation $\longrightarrow$ is *noetherian* if there exists no infinite chain

$$t_1 \longrightarrow t_2 \longrightarrow \cdots \longrightarrow t_i \longrightarrow \cdots,$$

and *confluent* if $s \stackrel{*}{\longrightarrow} t_1$ and $s \stackrel{*}{\longrightarrow} t_2$ imply that there is a t such that $t_1 \stackrel{*}{\longrightarrow} t$ and $t_2 \stackrel{*}{\longrightarrow} t$, The relation $\longrightarrow$ is a *rewrite relation* if $s \longrightarrow t$ implies that $u[s\theta] \longrightarrow u[t\theta]$ for all terms s, t and u, and substitutions θ, where $u[t]$ stands for u with certain subterm occurrence t.

Let E be a finite set of equations. We say that E is a *rewrite system* with respect to an ordering $\succ$ on terms if we have $s \succ t$ or $t \succ s$ for all equations $s \approx t$ in E. We sometimes write $E^{\succ}$ if E is a rewrite system with respect to $\succ$, to emphasize the ordering. An equation $s \approx t$ of E is a *rule* $s \to t$ of $E^{\succ}$ if $s \succ t$. By $\longrightarrow_{E^{\succ}}$ or simply $\longrightarrow_E$ we denote the smallest rewrite relation for which $s \longrightarrow_E t$ whenever $s \to t$ is a rule of E. We sometimes write $\longrightarrow$ for $\longrightarrow_E$ when E is clear from the context. A term s is said to be in *normal form* or *irreducible* with respect to E if there is no term t such that $s \longrightarrow_E t$. If a term t has a unique normal form with respect to E then this normal form is denoted by $t{\downarrow_E}$.

Let E be a rewrite system. Then E is *noetherian* if the corresponding rewrite relation $\longrightarrow_E$ is noetherian, and E is *confluent* if the corresponding rewrite relation $\longrightarrow_E$ is confluent. A rewrite system E is *convergent* or *canonical* if it is both noetherian and confluent. Convergent systems enjoy the property that each term has a unique normal form. Moreover, if we want to decide whether an equation $s \approx t$ logically follows from a set E of equations, and E is a convergent rewrite system, then it is enough to see if $s{\downarrow_E} = t{\downarrow_E}$ (cf [42, Section 2.4]), i.e.,

$$E \models s \approx t \quad \Leftrightarrow \quad s{\downarrow_E} = t{\downarrow_E}.$$

To construct a canonical rewrite system from a given set E of equations, while preserving the set of logical consequences of E, is the main motivation behind the completion procedure [86]. It is well-known that for any set of ground equations there exists an equivalent canonical rewrite system [92]. Moreover, such a system can be constructed in $O(n^3)$ time [51, 55] or even in $O(n \log n)$ time [138]. A simple property that guarantees that a ground rewrite system E is canonical is that it is *reduced* [138], i.e., for each rule $l \to r$ in E, l is irreducible with respect to $E \setminus \{l \to r\}$ and r is irreducible with respect to E.

It follows by Birkhoff's completeness theorem for equational logic [12] that, given a set of ground equations E and and a ground equation $s \approx t$, $s \approx t$ is a logical consequence of E iff s can be reduced to t by using the equations in E as rewrite rules in both directions.

2.4 FINITE TREE AUTOMATA

Finite tree automata, or simply tree automata from here on, are a generalization of classical automata. Tree automata were introduced, independently, in Doner [43] and Thatcher and Wright [143]. The main motivation was to obtain decidability results for the weak monadic second-order logic of the binary tree. Here we adopt the following definition of tree automata, based on rewrite rules [20, 22].

- A *tree automaton* or *TA* A is a quadruple (Q, Σ, R, F) where
  - Q is a finite set of constants called *states*,
  - Σ is a *signature* that is disjoint from Q,
  - R is a set of *rules* of the form $f(q_1, \ldots, q_n) \to q$, where $f \in \Sigma$ has arity $n \geq 0$ and $q, q_1, \ldots, q_n \in Q$,
  - $F \subseteq Q$ is the set of *final states*.

  A is called a *deterministic* TA or DTA if there are no two different rules in R with the same left-hand side.

Note that if A is deterministic then R is a reduced set of ground rewrite rules and thus canonical [138]. Tree automata as defined above are usually also called *bottom-up* tree automata. Acceptance for tree automata or recognizability is defined as follows.

- The set of terms *recognized* by a TA $A = (Q, \Sigma, R, F)$ is the set

$$T(A) = \{\, \tau \in \mathcal{T}_\Sigma \mid (\exists q \in F)\ \tau \xrightarrow{*}_R q \,\}.$$

  A set of terms is called *recognizable* if it is recognized by some TA.

Two tree automata are *equivalent* if they recognize the same set of terms. It is well-known that the nondeterministic and the deterministic versions of TAs have the same expressive power [43, 60, 143], i.e., for any TA there is an equivalent DTA. For an overview of the notion of recognizability in general algebraic structures see Courcelle [21] and the fundamental paper by Mezei and Wright [103].

2.5 CLASSICAL AUTOMATA THEORY

We use some notions from classical automata theory and follow Hopcroft and Ullman in that respect [76]. Characters are treated as constants, in the usual case.

Finite Automata

We use the following formal definition of a DFA.

- A *deterministic finite automaton* or *DFA* M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where
  - Q is a finite set of *states*,
  - Σ is a finite *input alphabet* disjoint from Q,
  - $\delta : Q \times \Sigma \to Q$ is the *transition function*,
  - $q_0 \in Q$ is the *initial* state, and
  - $F \subseteq Q$ is the set of *final states*.

The transition function can be *partial*, i.e., undefined for certain elements of $Q \times \Sigma$. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. The *language accepted by* M, denoted by $L(M)$, is the set of all strings $a_1 a_2 \ldots a_n \in \Sigma^*$, $n \geq 0$, such that there exists a sequence $q_1, q_2 \ldots, q_n$ of states such that q_n is a final state and $\delta(q_{j-1}, a_j) = q_j$ for $1 \leq j \leq n$.

Turing Machines

We use the following formal definition of a Turing machine.

- A *nondeterministic Turing machine* M is a 7-tuple

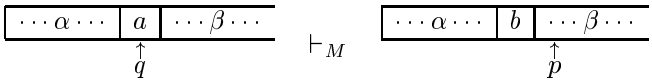
$$(Q, \Sigma_{\text{in}}, \Sigma, \delta, q_0, \text{ƀ}, F),$$

  where
  - Q is a finite set of *states*,
  - Σ is a finite set of *tape symbols* disjoint from Q,
  - $\text{ƀ} \in \Sigma$ is a tape symbol called *blank*,
  - Σ_{in} is a subset of Σ called the set of *input symbols*,
  - δ is a mapping from $Q \times \Sigma$ to subsets of $Q \times \Sigma \times \{\texttt{left}, \texttt{right}\}$, and is called the *transition function* of M,
  - q_0 is the *initial* state of M, and
  - $F \subseteq Q$ is the set of *final states*.

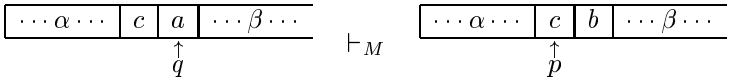
  M is *deterministic* if the range of δ consists of singelton sets, in which case we consider δ as a mapping from $Q \times \Sigma$ to $Q \times \Sigma \times \{\texttt{left}, \texttt{right}\}$.

An *instantaneous description* or *ID* of M is any string $\alpha q \beta$ where $q \in Q$ and $\alpha \in \Sigma^*$ and β is a string in Σ^* not ending with a blank. The intended meaning of an ID $\alpha q \beta$ of M is to give a complete description of a possible execution state of M: q is the state of the machine, α corresponds to the contents of the tape from the left edge of the tape to (but not including) the symbol pointed to by the tape head, and β is the rest of the contents of the tape terminated by the rightmost nonblank. So any "snapshot" of M during its computation is some ID (there can of course exist ID's that can never be reached by M).

A *move* is a pair (v, w) of ID's such that w follows from v according to the transition function of M. The binary relation of all moves of M is denoted by $\vdash_M$, and its transitive and reflexive closure by $\vdash_M^*$. The tape head can either move to the right: (assuming $(p, b, \texttt{right}) \in \delta(q, a)$)



or to the left: (assuming $(p, b, \texttt{left}) \in \delta(q, a)$)



The *language accepted by* M is the following set of strings:

$$L(M) = \{\, w \in \Sigma_{\text{in}}^* \mid q_0 w \vdash_M^* \alpha p \beta \text{ where } p \in F \text{ and } \alpha p \beta \text{ is an ID} \,\}.$$

The notions of valid (and invalid) computations [76] of a TM are a powerful tool in proving undecidability results about context free languages.

- A *valid computation* of M is a nonempty sequence $(w_1, w_2, \ldots, w_n)$ such that
  - each w_i is an ID of M, i.e., $w_i \in \Sigma^* Q (\Sigma^* \setminus \Sigma^* \text{ƀ})$ for $1 \leq i \leq n$,
  - w_1 is the *initial* ID, one of the form $q_0 v$ where $v \in \Sigma_{\text{in}}^*$,
  - w_n is a *final* ID, $w_n \in \Sigma^* F (\Sigma^* \setminus \Sigma^* \text{ƀ})$,
  - $w_i \vdash_M w_{i+1}$ for $1 \leq i < n$.

We use the following obvious relationship between valid computations and the language of M:

There is a valid computation of M with initial ID $q_0 v$ iff $v \in L(M)$.

Complexity Classes

We use the following computational complexity classes and corresponding acronyms in the thesis. In some cases there is no unanimous notation for a class in the literature. In particular, the class EXPTIME is denoted by DEXPTIME by some authors (to distinguish it from NEXPTIME) or by DEXP, and the class NL is sometimes denoted by NLOGSPACE or NLOG. Some authors prefer to write PTIME instead of P. We use the following shorthands.

NL The class of problems that can be solved nondeterministically within logarithmic space.

P The class of problems that can be solved deterministically within polynomial time.

NP The class of problems that can be solved nondeterministically within polynomial time.

PSPACE The class of problems that can be solved within polynomial space.

EXPTIME The class of problems that can be solved deterministically within exponential time, i.e., within time $2^{p(n)}$, where p is a polynomial and n the size of the input.

We refer the reader to Papadimitriou [112] and Johnson [81] for precise definitions and an extensive treatment of the subject. We establish some new P-completeness, PSPACE-completeness and EXPTIME-completeness results in the thesis. In the first case the reductions are within logarithmic space, and in the last two cases the reductions are in P, although we believe that all our reductions can be carried out within logarithmic space but we do not prove it formally. So we speak about P-completeness with respect to logarithmic space reductions and PSPACE-completeness or EXPTIME-completeness with respect to polynomial time reductions.

Undecidability of SREU

3.1 INTRODUCTION

The first undecidability proof of SREU was given by Degtyarev and Voronkov [34]. Before that result, there were several faulty proofs of its decidability, e.g. [52, 66]. In general, this quite unexpected undecidability result had a serious impact on the automated theorem proving community, as several effectively equivalent fundamental decision problems in automated reasoning in classical logic with equality turned out to be undecidable [30]. We return to this in the next chapter.

Here we show that *four* or even *three* rigid equations with *ground left-hand sides* and *two variables* in a signature with one binary function symbol an no other nonconstant function symbols, already imply undecidability. In fact, we give a uniform representation of all the recursively enumerable sets by using just three or four rigid equations with these properties. As a corollary we get that the undecidability of SREU holds already in very restricted cases.

At the end of this chapter we give a brief summary of the other proofs. The main idea behind our proof is based on a technique that was used by Plaisted [116] in a similar context, we refer to the technique as *shifted pairing* after Plaisted. The idea is to express repetition explicitly by a sequence of strings (like IDs of a TM). The first string of the sequence fulfills some initial conditions, the last string some final conditions and another sequence is used to check that the consequtive strings of the first sequence satisfy some relationship (like validity of a computation step).

A similar technique was used already by Goldfarb in the proof of the undecidability of second-order unification [64], which is by reduction of Hilbert's tenth problem, and later, adopted from that proof, also in a proof of the undecidability of SREU by Degtyarev and Voronkov [37], which is also by reduction of Hilbert's tenth problem. In this proof the key point is to explicitly represent the "history of a multiplication process".

We note also that shifted pairing bears certain similarities to the technique that is used to prove that any recursively enumerable set of strings is given by the intersection of two (deterministic) context free languages [76, Lemma 8.6].

3.2 OVERVIEW OF THE CONSTRUCTION

We consider a fixed Turing machine

$$M = (Q_M, \Sigma_{\text{in}}, \Sigma_{\text{tape}}, \delta, q_0, \text{ƀ}, \{q_{\text{acc}}\}),$$

and assume, without loss of generality, that the final ID of M is simply q_{acc} i.e., the tape is always empty when M enters the final state, and that $q_0 \neq q_{\text{acc}}$. Let also v be a string over the input alphabet of M. We effectively construct a system $S_v^M(x, y)$ of four rigid equations:

$$S_v^M(x,y) \quad = \quad \{\, S_{\text{id}}(x),\ S_{\text{mv}}(y),\ S_1(x,y),\ S_2(x,y)\,\}$$

where

$$\begin{aligned} S_{\text{id}}(x) &= E_{\text{id}} \models_\forall c'_{\text{id}} \centerdot x \approx c_{\text{id}}, \\ S_{\text{mv}}(y) &= E_{\text{mv}} \models_\forall c'_{\text{mv}} \centerdot y \approx c_{\text{mv}}, \\ S_1(x,y) &= \Pi_1 \models_\forall x \approx y, \\ S_2(x,y) &= \Pi_2 \models_\forall x \approx t_v \centerdot y \end{aligned}$$

where all the left-hand sides are ground, c'_{id}, c_{id}, c'_{mv} and c_{mv} are constants, '$\centerdot$' is the only nonconstant function symbol in the system and t_v is a ground term that represents the initial ID of M with input string v. We prove that M accepts v iff S_v^M is solvable. This establishes the undecidability result because all the steps in the construction are effective.

The main idea behind the rigid equations is roughly as follows. Assume that there is a substitution θ that solves the system.

- From θ being a solution of $S_{\text{id}}(x)$, it follows that $x\theta$ represents a sequence $(v_0, v_1, \ldots, v_m)$ of IDs of M, and v_m is the final ID of M.
- From θ being a solution of $S_{\text{mv}}(y)$, it follows that $y\theta$ represents a sequence
$$((w_0, w_0^+), (w_1, w_1^+), \ldots, (w_n, w_n^+))$$
of *moves* of M, i.e., $w_i \vdash_M w_i^+$ for $0 \leq i \leq n$.
- From θ being a solution of $S_1(x, y)$ it follows that $n = m$ and $v_i = w_i$ for $0 \leq i \leq m$.
- And finally, from θ being a solution of $S_2(x, y)$ it follows that $v_0 = v$ and $v_i = w_{i-1}^+$ for $1 \leq i \leq m$.

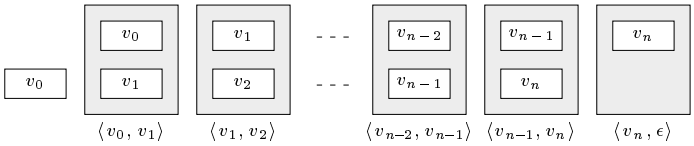


Figure 3.1: $(\langle v_0, v_1 \rangle, \langle v_1, v_2 \rangle, \ldots, \langle v_n, \epsilon \rangle)$ is a "shifted pairing" of $(v_0, v_1, \ldots, v_n)$.

The combination of the last two points is the so-called "shifted pairing" technique. This is illustrated by Figure 3.1. The outcome of this shifted pairing is that $x\theta$ is a valid computation of M with input v, and thus M accepts v. Conversely, if M accepts v then it is easy to construct a solution of the system. We now give a formal construction of the above idea.

3.3 WORDS AND TRAINS

Words are certain terms that we choose to represent strings with, and trains are certain terms that we choose to represent sequences of strings with. We use the letters v and w to stand for strings of constants. Let $\centerdot$ be a binary function symbol. We write it in infix notation and assume that it associates to the right. For example $t_1 \centerdot t_2 \centerdot t_3$ stands for the term $\centerdot(t_1, \centerdot(t_2, t_3))$.

- We say that a (ground) term t is a *c-word* if it has the form

$$a_1 \centerdot a_2 \centerdot \cdots \centerdot a_n \centerdot c$$

  for some $n \geq 0$ where each a_i and c is a constant. A *word* is a c-word for some constant c.

We use the following convenient shorthand notation for words. Let t be the word $a_1 \centerdot a_2 \centerdot \cdots \centerdot a_n \centerdot c$ and v the string $a_1 a_2 \cdots a_n$. We write $v \centerdot c$ for t and say that t *represents* v.

- *A* term t is called a *c-train* if it has the form

$$t_1 \centerdot t_2 \centerdot \cdots \centerdot t_n \centerdot c$$

  for some $n \geq 0$ where each t_i is a word and c is a constant. If $n = 0$ then t is said to be *empty*. The t_i's are called the *words of* t. A *train* is a c-train for some constant c.

By the *pattern* of a train

$$(v_1 \centerdot c_1) \centerdot (v_2 \centerdot c_2) \centerdot \cdots \centerdot (v_n \centerdot c_n) \centerdot c$$

we mean the string $c_1 c_2 \cdots c_n$. Let $\mathcal{V} = \{V_i\}_{i \in I}$ be a finite family of regular sets of strings over a finite set Σ of constants, where I is a set of constants disjoint from Σ. Let U be a regular set of strings over I and let c be a constant not in Σ or I.

- We let $\mathrm{Tn}(\mathcal{V}, U, c)$ denote the set of all c-trains t such that the pattern of t is in U and, for $i \in I$, each i-word of t represents a string in V_i.

Example 3.1 Consider the set $\mathrm{Tn}(\{V_a, V_b, V_c\}, ab^*c, \Lambda)$. This is the set of all Λ-trains t such that the first word of t is an a-word representing a string in V_a, the last word of t is a c-word representing a string in V_c and the middle ones (if any) are b-words representing strings in V_b. □

We say that a set of trains *has a regular pattern* if it is equal to some set $\mathrm{Tn}(\mathcal{V}, U, c)$ with $\mathcal{V}$, U and c as above. The main result of this section is the following theorem.

Theorem 3.1 (Train Theorem) *Any set of trains with a regular pattern is recognizable and a DTA that recognizes this set can be obtained effectively.*

The construction of the rigid equations S_{id} and S_{mv} follows easily from the Train Theorem. We believe that this theorem is of independent interest. For example, several theorems that are used in a similar context in Plaisted [116, Theorems 8.2–8.11], can be stated as corollaries of Theorem 3.1. Before we prove the theorem we state the following simple lemma. This lemma follows from the wellknown fact that all regular sets of strings are recognizable (cf [60]), assuming an appropriate representation of strings.[1] For any string v, we write v^{r} for v *in reverse* and for a set of strings V we let $V^{\mathrm{r}} = \{\, v^{\mathrm{r}} \mid v \in V \,\}$.

Lemma 3.1 *Let V be a regular set of strings over a set Σ of constants and c a constant not in Σ. Then $\{\, v \centerdot c \mid v \in V \,\}$ is recognizable and a DTA is obtained effectively from V.*

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts the reverse of V, or V^{r}, (clearly M exists, cf [76, p 281]). For each $a \in \Sigma$ let $\tilde{a}$ be a new state. Let A be the DTA (Q_A, Γ, R_A, F_A) where

$$\begin{aligned} Q_A &= Q \cup \{\, \tilde{a} \mid a \in \Sigma \,\}, \\ \Gamma &= \Sigma \cup \{\centerdot, c\}, \\ R_A &= \{\, \tilde{a} \centerdot q \to p \mid \delta(q, a) = p \,\} \cup \{\, a \to \tilde{a} \mid a \in \Sigma \,\} \cup \{c \to q_0\}, \\ F_A &= F. \end{aligned}$$

[1] Traditionally a string $a_1 a_2 \cdots a_n$ is represented by a term $a_n(\cdots a_2(a_1(q_0)))$, i.e., the symbols of the alphabet are treated as unary function symbols, and the term is written using the reverse notation $q_0 a_1 a_2 \cdots a_n$.

We must prove that, for all $t \in \mathcal{T}_\Gamma$,

$$t \xrightarrow{*}_{R_A} q \text{ for some } q \in F \quad \Leftrightarrow \quad t = v \centerdot c \text{ for some } v \in L(M)^{\mathrm{r}}.$$

Let us consider the direction '$\Leftarrow$' first. So assume that

$$v = a_{n-1}a_{n-2}\cdots a_0 \in L(M)^{\mathrm{r}},$$

i.e, $a_0 \cdots a_{n-2}a_{n-1} \in L(M)$. So, there exist $q_1, q_2 \ldots, q_n \in Q$, such that $q_n \in F$ and the following holds:

$$\delta(q_0, a_0) = q_1, \ldots, \delta(q_{n-2}, a_{n-2}) = q_{n-1},\ \delta(q_{n-1}, a_{n-1}) = q_n.$$

But then we can, by the definition of R_A, construct the following reduction:

$$\begin{aligned} v \centerdot c = a_{n-1}a_{n-2}\cdots a_0 \centerdot c &\xrightarrow{*} \tilde{a}_{n-1}\tilde{a}_{n-2}\cdots\tilde{a}_1\tilde{a}_0 \centerdot q_0 \\ &\longrightarrow \tilde{a}_{n-1}\tilde{a}_{n-2}\cdots\tilde{a}_1 \centerdot q_1 \\ &\xrightarrow{*} \tilde{a}_{n-1} \centerdot q_{n-1} \\ &\longrightarrow q_n \in F, \end{aligned}$$

which shows that $v \centerdot c \in T(A)$. The direction '$\Rightarrow$' follows also easily. First note that any term t in $\mathcal{T}_\Gamma$ that reduces to a final state q with respect to R_A must be a c-word that represents some string v over Σ. From the definition of R_A follows then, like above, that v must be in V. ⊠

We now prove the Train Theorem. For a more detailed proof see Veanes [147, Theorem 3.8].

Proof. Let $\mathcal{V}$, Σ, U, I, and c be like above. For each $i \in I$, let $\Sigma_i = \Sigma \cup \{\centerdot, i\}$ and let $A_i = (Q_i, \Sigma_i, R_i, F_i)$ be a DTA given by Lemma 3.1 such that

$$T(A_i) = \{\, v \centerdot i \mid v \in V_i \,\}.$$

Let $\Sigma_c = I \cup \{\centerdot, c\}$ and let $A_c = (Q_c, \Sigma_c, R_c, F_c)$ be a DTA given by Lemma 3.1 such that

$$T(A_c) = \{\, u \centerdot c \mid u \in U \,\}.$$

Assume, without loss of generality, that all the DTAs have mutually disjoint sets of states, except for the states $\tilde{a}$ for $a \in \Sigma$ that are the same in all the A_i's for $i \in I$. In fact, one can think of any constant $a \in \Sigma$ and the corresponding state $\tilde{a}$ *as being the same element.*

Let now R' be the set of rules obtained from R_c by relpacing, for all $i \in I$, each rule $\tilde{\imath} \centerdot p_1 \to p_2$ in it with the set of rules $\{\, q \centerdot p_1 \to p_2 \mid q \in F_i \,\}$, and discarding the rule $i \to \tilde{\imath}$. Let now R be the following set of rules:

$$R = \bigcup_{i \in I} R_i \cup R'.$$

Note that R is a reduced set of rewrite rules due to the disjointness assumptions and the assumption that the states $\tilde{a}$ for $a \in \Sigma$ are the same in all the DTAs. We are now ready to define A as the DTA (Q, Γ, R, F_c) where $\Gamma = \Sigma \cup I \cup \{\centerdot, c\}$ and

$$Q = \bigcup_{i \in I} Q_i \cup (Q_c \setminus \{\, \tilde{i} \mid i \in I \,\}).$$

We can now prove that

$$T(A) = \mathrm{Tn}(\mathcal{V}, U, c).$$

Let use consider the direction '$\subseteq$' first. Assume that $t \in T(A)$, i.e., t reduces to some state q in F_c via the rules in R. This reduction is only possible if it has (in principle) the following form:[2]

$$t \xrightarrow{*}_R q_1 q_2 \cdots q_n \centerdot c \xrightarrow{*}_{R'} q.$$

where each q_k is in F_{i_k} for some $i_k \in I$. Furthermore, by definition of R' and A_c, we know that $i_1 i_2 \cdots i_n \in U$. The first part of the reduction is possible only if

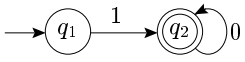
$$t = t_1 \centerdot t_2 \centerdot \cdots \centerdot t_n \centerdot c,$$

where each t_k reduces to q_k. Note that, due to the disjointness properties of the DTAs, only the rules in R_{i_k} can be used in the reduction $t_k \xrightarrow{*} q_k$, and thus $t_k \in T(A_{i_k})$. Hence each t_k has the form $v \centerdot i_k$ for some $v \in V_{i_k}$, and the pattern of t is $i_1 i_2 \cdots i_n$, which we know is in U. This proves that $t \in \mathrm{Tn}(\mathcal{V}, U, c)$.

Let us now consider the direction '$\supseteq$'. So assume that $t = t_1 \centerdot t_2 \centerdot \cdots \centerdot t_n \centerdot c$ where each t_k is in $T(A_{i_k})$ for some $i_k \in I$ and $i_1 i_2 \cdots i_n \in U$. It follows that each t_k reduces with R_{i_k} to some $q_k \in F_{i_k}$ and thus t reduces to $q_1 q_2 \cdots q_n \centerdot c$. By definition of R', $q_1 q_2 \cdots q_n \centerdot c$ reduces to some $q \in F_c$. It follows that $t \xrightarrow{*}_R q$ for some $q \in F_c$ and thus $t \in T(A)$. ⊠

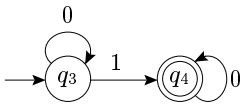
The following example illustrates the construction in the proof of the Train Theorem.

Example 3.2 Let $\Sigma = \{0, 1\}$, $I = \{\mathtt{a}, \mathtt{b}\}$ and let Λ be a new constant. Let $\mathcal{V} = \{V_i\}_{i \in I}$ where $V_{\mathtt{a}} = \mathtt{0^*1}$ and $V_{\mathtt{b}} = \mathtt{0^*10^*}$. Let $U = \mathtt{bab^*a}$. We construct a DTA that recognizes the set $\mathrm{Tn}(\mathcal{V}, U, \Lambda)$. Consider the following transition diagrams of a DFA for $V_{\mathtt{a}}{}^{\mathrm{r}}$:



[2] A formal argument can be given by using induction and proving some lemmas first [147, Chapter 3].

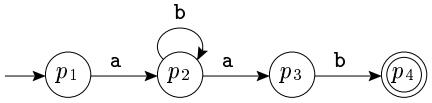
and of a DFA for $V_{\mathtt{b}}{}^{\mathrm{r}}$:



By following the construction in Lemma 3.1 we get that the rules of $A_{\mathtt{a}}$ and $A_{\mathtt{b}}$ are as follows:

$$\begin{aligned} R_{\mathtt{a}} &= \{\mathtt{1} \to \tilde{\mathtt{1}},\ \mathtt{0} \to \tilde{\mathtt{0}},\ \mathtt{a} \to q_1,\ \tilde{\mathtt{1}}\,\mathtt{.}\,q_1 \to q_2,\ \tilde{\mathtt{0}}\,\mathtt{.}\,q_2 \to q_2\}, \\ R_{\mathtt{b}} &= \{\mathtt{1} \to \tilde{\mathtt{1}},\ \mathtt{0} \to \tilde{\mathtt{0}},\ \mathtt{b} \to q_3,\ \tilde{\mathtt{0}}\,\mathtt{.}\,q_3 \to q_3,\ \tilde{\mathtt{1}}\,\mathtt{.}\,q_3 \to q_4,\ \tilde{\mathtt{0}}\,\mathtt{.}\,q_4 \to q_4\}. \end{aligned}$$

For the set U^{r} we can consider a DFA with the following transition diagram:



From this we can extract the DTA A_Λ with the following set of rules:

$$\begin{aligned} R_\Lambda &= \{\mathtt{a} \to \tilde{\mathtt{a}},\ \mathtt{b} \to \tilde{\mathtt{b}},\ \Lambda \to p_1, \\ &\quad \tilde{\mathtt{a}}\,\mathtt{.}\,p_1 \to p_2,\ \tilde{\mathtt{b}}\,\mathtt{.}\,p_2 \to p_2,\ \tilde{\mathtt{a}}\,\mathtt{.}\,p_2 \to p_3,\ \tilde{\mathtt{b}}\,\mathtt{.}\,p_3 \to p_4\}. \end{aligned}$$

Now, following the construction in the Train Theorem, we get that the DTA A has the following set of rules. First, a set R' is constructed by removing the first two rules in R_Λ and replacing $\tilde{\mathtt{a}}$ and $\tilde{\mathtt{b}}$ with q_2 and q_4, respectively. Second, R is taken as the union of $R_{\mathtt{a}}$, $R_{\mathtt{b}}$ and R' . So R is the following set of rules:

$$\begin{aligned} R &= \{\mathtt{1} \to \tilde{\mathtt{1}},\ \mathtt{0} \to \tilde{\mathtt{0}},\ \mathtt{a} \to q_1,\ \tilde{\mathtt{1}}\,\mathtt{.}\,q_1 \to q_2,\ \tilde{\mathtt{0}}\,\mathtt{.}\,q_2 \to q_2\} \cup \\ &\quad \{\mathtt{b} \to q_3,\ \tilde{\mathtt{0}}\,\mathtt{.}\,q_3 \to q_3,\ \tilde{\mathtt{1}}\,\mathtt{.}\,q_3 \to q_4,\ \tilde{\mathtt{0}}\,\mathtt{.}\,q_4 \to q_4\} \cup \\ &\quad \{\Lambda \to p_1,\ q_2\,\mathtt{.}\,p_1 \to p_2,\ q_4\,\mathtt{.}\,p_2 \to p_2,\ q_2\,\mathtt{.}\,p_2 \to p_3,\ q_4\,\mathtt{.}\,p_3 \to p_4\}. \end{aligned}$$

Let us consider a reduction in R. Let us write a Λ-train $t_1\,\mathtt{.}\,t_2\,\mathtt{.}\cdots\mathtt{.}\,t_n\,\mathtt{.}\,\Lambda$ as $[t_1, t_2, \ldots, t_n]$. Take for example

$$t = [\mathtt{010.b},\quad \mathtt{001.a},\quad \mathtt{1.b},\quad \mathtt{01.a}].$$

The pattern of t is `baba` which is in U. Let us see how t reduces to p_4.

$$\begin{aligned} t &\xrightarrow{*}_R [\tilde{\mathtt{0}}\tilde{\mathtt{1}}\tilde{\mathtt{0}}\,\mathtt{.}\,q_3,\quad \tilde{\mathtt{0}}\tilde{\mathtt{0}}\tilde{\mathtt{1}}\,\mathtt{.}\,q_1,\quad \tilde{\mathtt{1}}\,\mathtt{.}\,q_3,\quad \tilde{\mathtt{0}}\tilde{\mathtt{1}}\,\mathtt{.}\,q_1] \\ &\xrightarrow{*}_R [\tilde{\mathtt{0}}\tilde{\mathtt{1}}\,\mathtt{.}\,q_3,\quad \tilde{\mathtt{0}}\tilde{\mathtt{0}}\,\mathtt{.}\,q_2,\quad q_4,\quad \tilde{\mathtt{0}}\,\mathtt{.}\,q_2] \\ &\xrightarrow{*}_R [\tilde{\mathtt{0}}\,\mathtt{.}\,q_4,\quad \tilde{\mathtt{0}}\,\mathtt{.}\,q_2,\quad q_4,\quad q_2] \\ &\xrightarrow{*}_R [q_4,\quad q_2,\quad q_4,\quad q_2] \\ &\longrightarrow_{R'} q_4q_2q_4q_2\,\mathtt{.}\,p_1 \\ &\xrightarrow{*}_{R'} q_4. \end{aligned}$$

□

3.4 REPRESENTING IDS AND MOVES

In this section we show how to construct the rigid equations $S_{\mathrm{id}}(x)$ and $S_{\mathrm{mv}}(y)$. Our main tool is the Train Theorem. In doing so, we use the following observation, that relates rigid E-unification with recognizability.

Lemma 3.2 *Let $A = (Q, \Sigma, R, F)$ be a DTA, f a binary function symbol, and c_1 and c_2 constants not in Q or Σ. There is a set of ground equations E such that for all θ with range $\mathcal{T}_\Sigma$, θ solves $E \models_{\forall} f(c_1, x) \approx c_2$ iff $x\theta \in T(A)$.*

Proof. Let $E = R \cup \{ f(c_1, q) \to c_2 \mid q \in F \}$. It is easy to check that E is a reduced rewrite system and thus canonical, and since c_2 is irreducible with respect to E we have in particular for all $t \in \mathcal{T}_\Sigma$ that

$$\begin{aligned} E \models f(c_1, t) \approx c_2 \quad &\Leftrightarrow \quad f(c_1, t) \xrightarrow{*}_E c_2 \\ &\Leftrightarrow \quad t \xrightarrow{*}_R q \text{ for some } q \in F \end{aligned}$$

and so the statement follows. ⊠

Let us assign arity 0 to all the tape symbols (Σ_{tape}) and all the states (Q_M) of M. Let Σ be the following signature:

$$\Sigma = \Sigma_{\mathrm{tape}} \cup Q_M \cup \{e_0, e_1, \Lambda, .\},$$

where e_0, e_1 and Λ are new constants.

Representing ID Sequences

Recall that an *ID* of M is any string in $\Sigma^*_{\mathrm{tape}} Q_M \Sigma^*_{\mathrm{tape}}$ that does not end with a blank ($\flat$). We represent IDs by e-words, where e is one of e_0 or e_1. In particular, the final ID is represented by the word $q_{\mathrm{acc}} \,.\, e_1$ and IDs in general are represented by corresponding e_0-words.

- Any train of the form

  $$(v_0 \,.\, e_0) \,.\, (v_1 \,.\, e_0) \,.\, (v_2 \,.\, e_0) \,.\, \cdots \,.\, (v_n \,.\, e_0) \,.\, (q_{\mathrm{acc}} \,.\, e_1) \,.\, \Lambda,$$

  where $n \geq 0$ and each v_i is an ID of M, is called an *ID-train*.

It is clear that the set of all IDs and the set consisting of just the final ID are regular sets. The set of patterns of the ID-trains is given by the regular expression $e_0 e_0^* e_1$. By using the Train Theorem, let

$$A_{\mathrm{id}} = (Q_{\mathrm{id}}, \Sigma, R_{\mathrm{id}}, F_{\mathrm{id}})$$

be a DTA that recognizes the set of all ID-trains. Let c'_{id} and c_{id} be two new constants and, by using Lemma 3.2, let $S_{\mathrm{id}}(x)$ be such that,

$$\{ x\theta \in \mathcal{T}_\Sigma \mid \theta \text{ solves } S_{\mathrm{id}}(x) \} = T(A_{\mathrm{id}}).$$

Representing Move Sequences

Let c_{ab} be a new constant for each pair of constants a and b in the set $\Sigma_{\text{tape}} \cup Q_M$. Let also e_2 and Λ' be new constants. Let now Γ be the following signature:

$$\Gamma = \{\, c_{ab} \mid a, b \in \Sigma_{\text{tape}} \cup Q_M \,\} \cup \{e_2, \Lambda', .\}$$

Note that $.$ is the only symbol that occurs in both Σ and Γ.

For and ID w of M we let w^+ denote the successor of w with respect to the transition function of M. For technical reasons it is convenient to let $q^+_{\text{acc}} = \epsilon$, i.e., the successor of the final ID is the empty string. The pair (w, w^+) is called a *move*. Let $w = a_1 a_2 \cdots a_m$ and $w^+ = b_1 b_2 \cdots b_n$ for some $m \geq 1$ and $n \geq 0$. Note that $n \in \{m-1, m, m+1\}$. Let $k = \max(m, n)$. If $m < n$ let $a_k = \text{ƀ}$ and if $n < m$ let $b_k = \text{ƀ}$, i.e., pad the shorter of the two strings with a blank at the end.

- We write $\langle w, w^+ \rangle$ for the string $c_{a_1 b_1} c_{a_2 b_2} \cdots c_{a_k b_k}$ and say that the e_2-word $\langle w, w^+ \rangle . e_2$ *represents* the move (w, w^+). By a *move-train* we mean any Λ'-train
$$t = t_0 . t_1 . \cdots . t_n . \Lambda',$$
such that each t_i represents a move and $n \geq 1$.

Example 3.3 Take $\Sigma_{\text{in}} = \{\mathtt{0}, \mathtt{1}\}$, and let $\mathtt{q}, \mathtt{p} \in Q_M$. Assume that the transition function δ of M is such that, when the tape head points to a blank and the state is $\mathtt{q}$ then a $\mathtt{1}$ is written to the tape, the tape head moves left and M enters state $\mathtt{p}$, i.e., $\delta(\mathtt{q}, \text{ƀ}) = (\mathtt{p}, \mathtt{1}, \mathtt{L})$. Imagine that the current ID is $\mathtt{00q}$, i.e., the tape contains the string $\mathtt{00}$ and the tape head points to the bank following the last $\mathtt{0}$. So $(\mathtt{00q}, \mathtt{0p01})$ is a move. This move is represented by the word $c_{\mathtt{00}} . c_{\mathtt{0p}} . c_{\mathtt{q0}} . c_{\text{ƀ}\mathtt{1}} . e_2$, i.e., $\langle \mathtt{00q}, \mathtt{0p01} \rangle . e_2$. □

It is straightforward to see that the set of all strings $\langle w, w^+ \rangle$ where w is an ID, is a regular set. The patterns of all move-trains are given by the regular expression $e_2 e_2 e_2^*$. By using the Train Theorem let

$$A_{\text{mv}} = (Q_{\text{mv}}, \Gamma, R_{\text{mv}}, F_{\text{mv}})$$

be a DTA that recognizes the set of all move-trains. Assume also that Q_{mv} and Q_{id} are disjoint. Let c'_{mv} and c_{mv} be new constants and, by using Lemma 3.2, let $S_{\text{mv}}(y)$ be such that,

$$\{\, y\theta \in \mathcal{T}_\Gamma \mid \theta \text{ solves } S_{\text{mv}}(y) \,\} = T(A_{\text{mv}}).$$

3.5 FINAL CONSTRUCTION

In this section we finish the construction of S_v^M and prove the undecidability results. The only essential components that we have not defined yet are Π_1 and Π_2. We let Π_1 and Π_2 be the (sets of equations corresponding to the) following rewrite systems. The differences between Π_1 and Π_2 are indicated with frames.

$$\begin{aligned}
\Pi_1 &= \{\, c_{ab} \to \boxed{a} \mid a, b \in \Sigma_{\text{tape}} \cup Q_M \,\} \cup \\
&\quad \{\, e_1 \to e_0,\ e_2 \to e_0,\ \Lambda' \to \Lambda,\ \flat \centerdot e_0 \to e_0 \,\} \\
\Pi_2 &= \{\, c_{ab} \to \boxed{b} \mid a, b \in \Sigma_{\text{tape}} \cup Q_M \,\} \cup \\
&\quad \{\, e_1 \to e_0,\ e_2 \to e_0,\ \Lambda' \to \Lambda,\ \flat \centerdot e_0 \to e_0,\ \boxed{e_0 \centerdot \Lambda \to \Lambda} \,\}
\end{aligned}$$

It is easy to see that both sets are in fact reduced sets of ground rewrite rules and thus canonical. We can now state the main theorem of this section. For any input string v for M let the term t_v in the system S_v^M be the word $q_0 v \centerdot e_0$, i.e., t_v represents the initial ID of M with input v.

Theorem 3.2 *$S_v^M(x, y)$ is solvable iff M accepts v.*

Before proving the theorem we state and prove some useful lemmas.

Lemma 3.3 *If θ solves $S_1(x, y)$ and $S_2(x, y)$ then $x\theta, y\theta \in \mathcal{T}_{\Sigma\cup\Gamma}$.*

Proof. We prove by induction on the size of $x\theta$ that if θ solves the following system, where t_0 is any term in $\mathcal{T}_{\Sigma\cup\Gamma}$, then $x\theta, y\theta \in \mathcal{T}_{\Sigma\cup\Gamma}$.

$$\{\, \Pi_1 \nvDash x \approx y, \quad \Pi_2 \nvDash x \approx t_0 \centerdot y \,\}$$

The statement follows then by choosing $t_0 = q_0 v \centerdot e_0$.

So consider a fixed t_0 and assume that θ solves the above system. If $x\theta$ is a constant then so is its normal form in Π_2, say $x\theta{\downarrow_{\Pi_2}} = c$, and so $t_0 \centerdot y\theta \xrightarrow{*}_{\Pi_2} c$. But then $c \in \Sigma$ and consequently $x\theta, y\theta \in \mathcal{T}_{\Sigma\cup\Gamma}$. The cases when $x\theta$ is not a constant, but either $x\theta{\downarrow_{\Pi_1}}$ or $x\theta{\downarrow_{\Pi_2}}$ is a constant, are also immediate.

So assume that $x\theta = t_1 \centerdot t$ and $(t_1 \centerdot t){\downarrow_{\Pi_i}} = t_1{\downarrow_{\Pi_i}} \centerdot t{\downarrow_{\Pi_i}}$ for $i \in \{1, 2\}$. So $t_1{\downarrow_{\Pi_2}} = t_0{\downarrow_{\Pi_2}}$ and thus $t_1 \in \mathcal{T}_{\Sigma\cup\Gamma}$ since $t_0 \in \mathcal{T}_{\Sigma\cup\Gamma}$; also

$$\Pi_2 \models t \approx y\theta.$$

It follows from $\Pi_1 \models t_1 \centerdot t \approx y\theta$ that $y\theta = s_1 \centerdot s$ for some terms s_1 and s such that

$$\Pi_1 \models t \approx s$$

and $\Pi_1 \models s_1 \approx t_1$. From the latter follows that $s_1 \in \mathcal{T}_{\Sigma\cup\Gamma}$ because $t_1 \in \mathcal{T}_{\Sigma\cup\Gamma}$. Let now θ' be such that $x\theta' = t$ and $y\theta' = s$. So θ' solves the system

$$\{\, \Pi_1 \models_\forall x \approx y, \quad \Pi_2 \models_\forall x \approx s_1 \centerdot y \,\},$$

and it follows by the induction hypothesis that t and s are in $\mathcal{T}_{\Sigma\cup\Gamma}$, and consequently, so are $t_1 \centerdot t = x\theta$ and $s_1 \centerdot s = y\theta$. ⊠

Lemma 3.4 *If θ solves $S_v^M(x,y)$ then $x\theta \in \mathcal{T}_\Sigma$ and $y\theta \in \mathcal{T}_\Gamma$.*

Proof. Assume that θ solves $S_v^M(x,y)$. Obviously $x\theta \in \mathcal{T}_{\Sigma\cup Q_{\rm id}}$ since θ solves $S_{\rm id}(x)$. By Lemma 3.3 we know also that $x\theta \in \mathcal{T}_{\Sigma\cup\Gamma}$. But $\Sigma \setminus \{\centerdot\}$, $\Gamma \setminus \{\centerdot\}$ and $Q_{\rm id}$ are mutually disjoint, and thus $x\theta \in \mathcal{T}_\Sigma$. A similar argument shows that $y\theta \in \mathcal{T}_\Gamma$. ⊠

Lemma 3.5 *If θ solves $S_v^M(x,y)$ then $x\theta$ is an ID-train and $y\theta$ is a move-train.*

Proof. Assume that θ solves $S_v^M(x,y)$. By Lemma 3.4 follows that $x\theta \in \mathcal{T}_\Sigma$ and $y\theta \in \mathcal{T}_\Gamma$. The statement follows now by the definition of the DTAs $A_{\rm id}$ and $A_{\rm mv}$. ⊠

We can now prove Theorem 3.2.

Proof. We prove that $S_v^M(x,y)$ is solvable $\quad\Leftrightarrow\quad$ M accepts v.

Proof of '⇒' Let θ be a substitution that solves $S_v^M(x,y)$. By using Lemma 3.5 we get that $x\theta$ and $y\theta$ have the following form:

$$\begin{aligned} x\theta &= (v_0 \centerdot e_0) \centerdot (v_1 \centerdot e_0) \centerdot \cdots \centerdot (v_{m-1} \centerdot e_0) \centerdot (v_m \centerdot e_1) \centerdot \Lambda \\ y\theta &= (\langle w_0, w_0^+\rangle \centerdot e_2) \centerdot (\langle w_1, w_1^+\rangle \centerdot e_2) \centerdot \cdots \centerdot (\langle w_n, w_n^+\rangle \centerdot e_2) \centerdot \Lambda' \end{aligned}$$

where $m \geq 1$, $n \geq 1$ and all the v_i's and w_i's are IDs of M and $v_m = q_{\rm acc}$. Since θ solves $S_1(x,y)$, it follows that the normal forms of $x\theta$ and $y\theta$ under Π_1 must coincide. But

$$\begin{aligned} x\theta{\downarrow_{\Pi_1}} &= (v_0 \centerdot e_0) \centerdot (v_1 \centerdot e_0) \centerdot \cdots \centerdot (v_{m-1} \centerdot e_0) \centerdot (v_m \centerdot e_0) \centerdot \Lambda, \\ y\theta{\downarrow_{\Pi_1}} &= (w_0 \centerdot e_0) \centerdot (w_1 \centerdot e_0) \centerdot \cdots \centerdot (w_{n-1} \centerdot e_0) \centerdot (w_n \centerdot e_0) \centerdot \Lambda. \end{aligned}$$

Note that each term $\langle w_i, w_i^+\rangle \centerdot e_2$ reduces first to $w_i' \centerdot e_0$ where $w_i' = w_i$ or $w_i' = w_i\text{ƀ}$. The extra blank at the end is removed with the rule $\text{ƀ} \centerdot e_0 \to e_0$. So

$$n = m, \quad v_n = q_{\rm acc}, \quad v_i = w_i \ (0 \leq i \leq n). \tag{3.1}$$

Since θ solves $S_2(x,y)$ it follows that the normal forms of $x\theta$ and $(q_0 v \centerdot e_0)\centerdot y\theta$ under Π_2 must coincide. But

$$x\theta{\downarrow_{\Pi_2}} = x\theta{\downarrow_{\Pi_1}}$$

because $x\theta$ does not contain any constants from Γ and the rule $e_0 \centerdot \Lambda \to \Lambda$ is not applicable. Moreover, since $w_n = q_{\rm acc}$, it follows that $w_n^+ = \epsilon$ and thus $\langle w_n, w_n^+\rangle \centerdot e_0 = c_{q_{\rm acc}\flat} \centerdot e_0$. But

$$(c_{q_{\rm acc}\flat} \centerdot e_0) \centerdot \Lambda \longrightarrow\!\!\!\!\!\rightarrow_{\Pi_2} (\flat \centerdot e_0) \centerdot \Lambda \longrightarrow\!\!\!\!\!\rightarrow_{\Pi_2} e_0 \centerdot \Lambda \longrightarrow\!\!\!\!\!\rightarrow_{\Pi_2} \Lambda.$$

The normal form of $(q_0 v \centerdot e_0) \centerdot y\theta$ under Π_2 is thus

$$(q_0 v \centerdot e_0) \centerdot (w_0^+ \centerdot e_0) \centerdot (w_1^+ \centerdot e_0) \centerdot \cdots \centerdot (w_{n-1}^+ \centerdot e_0) \centerdot \Lambda.$$

It follows that $v_0 = q_0 v$, i.e., v_0 is the initial ID of M with input v, and

$$w_i^+ = v_{i+1} \ (0 \leq i < n). \tag{3.2}$$

From (3.1) and (3.2) follows now that $(v_0, v_1, \ldots, v_n)$ is a valid computation of M, and thus M accepts v.

Proof of '$\Leftarrow$' Assume that M accepts v. So there exists a valid computation $(v_0, v_1, \ldots, v_n)$ of M where $v_0 = q_0 v$, $v_n = q_{\rm acc}$ and $v_i^+ = v_{i+1}$ for $0 \leq i < n$. Let θ be such that $x\theta$ is the corresponding ID-train and $y\theta$ the corresponding move-train. It follows easily that θ solves $S_M(x,y)$. ⊠

The *shifted pairing* technique that is used in Theorem 3.2 is illustrated in Figure 3.1. So the following result is an immediate consequence of Theorem 3.2, because all the constructions involved with it are effective.

Corollary 3.1 (Degtyarev–Voronkov) *SREU is undecidable.*

Furthermore, the following result (due to Plaisted [116]) is an immediate consequence.

Corollary 3.2 (Plaisted) *SREU is undecidable already if the left-hand sides are ground.*

Furthermore, we can sharpen this result as follows.

Corollary 3.3 *SREU is undecidable if the left-hand sides are ground, there are only two variables and four rigid equations and one binary function symbol.*

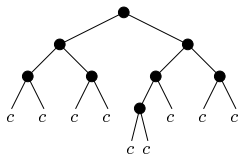


Figure 3.2: The term $((c \cdot c) \cdot (c \cdot c)) \cdot (((c \cdot c) \cdot c) \cdot (c \cdot c))$.

The undecidability with two variables and four rigid equations seems like an artifical extra condition, but in fact, it turns out to be an important special case. One implication is that the provability problem for the $\exists\exists$-fragment of intuitionistic logic with equality is undecidable (see Chapter 7). Another important fact is that two variables are *necessary* to get undecidability. If there is only *one* variable then SREU is decidable (see Chapter 6). In the next section we show that already *three* rigid equations are enough to imply undecidability. The case with *two* rigid equations is one of the few remaining open problems.

Remark We can also note that one constant is already enough. One can easily simulate any number of constants with just one constant c and $\cdot$, e.g., as follows. Assume that we need at most 2^k constants for some positive integer k. Then the i'th constant can be simulated by the term correspoding to the perfectly balanced binary tree of depth $k+1$ and with 2^k+1 leaves such that the i'th vertex at level k is internal and all the others are external. For example if $k=3$ then the fifth simulated constant would be the term in Figure 3.2. It is easy to see that the above theorems and proofs remain intact if each constant is replaced by the corresponding simulated constant.

3.6 MINIMAL CASE OF UNDECIDABILITY OF SREU

In this section we show that the two DTAs A_{id} and A_{mv} above can be combined into one DTA A by using elementary techniques of finite tree automata theory. By this way we reduce the number of rigid equations in S_v^M into *three* and obtain a sharper version of Corollary 3.3.

Let $A = (Q', \Sigma', R', F')$ be the the following DTA, where q is a new state,

$$\begin{aligned} Q' &= Q_{\mathrm{id}} \cup Q_{\mathrm{mv}} \cup \{q\}, \\ \Sigma' &= \Sigma \cup \Gamma, \\ R' &= R_{\mathrm{id}} \cup R_{\mathrm{mv}} \cup \{\, q_1 \cdot q_2 \to q \mid q_1 \in F_{\mathrm{id}},\ q_2 \in F_{\mathrm{mv}} \,\}, \\ F' &= \{q\}. \end{aligned}$$

By the disjointness conditions between A_{id} and A_{mv} it follows that A is indeed a deterministic tree automaton, and thus R a canonical rewrite system. It follows by elementary properties of tree automata that

$$T(A) = \{\, t \centerdot s \mid t \in T(A_{\mathrm{id}}),\ s \in T(A_{\mathrm{mv}}) \,\}.$$

Let now $\hat{S}_v^M(x,y)$ be the following system of rigid equations:

$$\hat{S}_v^M(x,y) \quad = \quad \{R' \models_\forall x \centerdot y \approx q,\ S_1(x,y),\ S_2(x,y)\}.$$

We can now prove the following result.

Theorem 3.3 *$\hat{S}_v^M(x,y)$ is solvable iff M accepts v.*

Proof. By Theorem 3.2 it is enough to prove that for all θ,

$$\theta \text{ solves } \hat{S}_v^M(x,y) \quad \Leftrightarrow \quad \theta \text{ solves } S_v^M(x,y).$$

By Lemma 3.3 follows that we only need to consider θ such that $x\theta, y\theta \in \mathcal{T}_{\Sigma'}$. But, for all such θ,

$$\begin{aligned} R' \models x\theta \centerdot y\theta \approx q \quad &\Leftrightarrow \quad x\theta \centerdot y\theta \xrightarrow{*}_{R'} q \\ &\Leftrightarrow \quad x\theta \centerdot y\theta \in T(A) \\ &\Leftrightarrow \quad x\theta \in T(A_{\mathrm{id}}) \text{ and } y\theta \in T(A_{\mathrm{mv}}) \\ &\Leftrightarrow \quad \theta \text{ solves } S_{\mathrm{id}}(x) \text{ and } S_{\mathrm{mv}}(y). \end{aligned}$$

The rest is obvious. ⊠

We can note that the above construction is very general, since the choice of M and v is arbitrary. In particular, we can choose as M a *universal* Turing machine M_{u}. Let for example M_{u} be the Turing machine that accepts the universal language L_{u} [76, Section 8.3],

$$L_{\mathrm{u}} = \{\, \langle M, v\rangle \mid M \text{ is a Turing machine that accepts } v \,\},$$

where $\langle M, v\rangle$ is some encoding of the pair (M, v) that is carried out in some fixed alphabet. The precise details of such an encoding are not relevant here. We can now use the observation that the construction of R', Π_1 and Π_2 in $\hat{S}_v^{M_{\mathrm{u}}}$ is independent of v and let $\hat{S}^{\mathrm{u}}(z,x,y)$ be the system $\hat{S}_v^{M_{\mathrm{u}}}(x,y)$ but with t_v replaced by the variable z. So, for any Turing machine M and input string v we have, by Theorem 3.3, that

$$\hat{S}^u(t_{\langle M,v\rangle}, x, y) \text{ is solvable} \quad \Leftrightarrow \quad M \text{ accepts v}$$

We conclude with the following sharpening of Corollary 3.3.

Corollary 3.4 *SREU is undecidable already for some fixed ground left-hand sides, two variables and three rigid equations.*

3.7 UNDECIDABILITY PROOFS OF SREU

In this section we briefly summarize the main points in the other undecidability proofs of SREU that have emerged since the problem was first [34] found to be undecidable. The different proofs reflect the undecidable nature of SREU more or less directly. The most transparent proof is probably by reduction of second-order unification, which shows how closely these problems are related. The proof by reduction of Hilbert's tenth problem is less transparent, but reveals that one can express certain derivations with a system of rigid equations.

Reduction of Monadic Semi-unification

The first proof of the undecidability of SREU [34] was by reduction of the monadic semi-unification problem to SREU. This proof has its roots in [31] where it is proved that the variable-bounded semi-unification problem[3] can be reduced to SREU. Semi-unification was proved undecidable in Kfoury, Tiuryn and Urzyczyn [85] and the monadic semi-unification was proved undecidable in Baaz [2]. A *semi-unification problem* consists of a set of expressions $s_i \leq t_i$, $1 \leq i \leq n$, where s_i and t_i are terms. Its solution consists of a substitution σ and a set of substitutions τ_i, $1 \leq i \leq n$, such that $\tau_i \sigma s_i$ coincides with σt_i. In the *monadic* case each τ_i is either empty or involves exactly one variable.

The first step in reducing the monadic semi-unification to SREU is to give a uniform (in n) presentation of this problem by a finite set of (simpler) Φ-unification problems. A Φ-unification problem corresponds roughly to some particular permutation (or guess) of n variables invoved in the τ_i's (there are at most $n!$ such guesses). It follows that Φ-unification is undecidable. A Φ-unification problem is then reduced to SREU. This reduction is rather technical, and it does not really reveal the reasons why SREU is undecidable.

Reduction of Second-Order Unification

The second proof of the undecidability of SREU by Degtyarev and Voronkov [33, 38], and probably the most straightforward one, is by reducing second-order unification to SREU. The undecidability of second-order unification was proved by Goldfarb [64].

A *second-order unification problem* is the problem of deciding if a finite set S of second-order equations is unifiable. A *second-order equation* is an expression $t \approx s$ where t and s are terms with possibly some (second-order) variables in place of function symbols. One can assume, without loss of generality, that all the equations in S are such that

[3] Decidability of the variable-bounded semi-unification problem is an open problem.

1. either all variables in t and s are first-order, or
2. that $s = X(s_1, \ldots, s_m)$ where all variables in all the s_i and t are first-order and X is a second-order variable.

In the second case a second-order substitution θ maps X to a term $X\theta$ where the so-called "bound" variables $w_1, \ldots, w_m$ (say $\vec{w}$) may occur, meaning that $X\theta$ corresponds to the λ-abstraction $\lambda\vec{w}.X\theta$. Now, θ is a unifier of $s \approx t$ iff it is the case that if we replace w_i in $X\theta$ by $s_i\theta$, for $1 \leq i \leq m$, then we obtain $t\theta$.

The set S is reduced (roughly) to the following system of rigid equations [38, Theorem 1]. The first case is simply reduced to the (rigid) equation $\models_\forall t \approx s$. The second case is reduced to two rigid equations, the first one stating that X is a term possibly containing new "constants" from $\vec{w}$, the second one stating that $\{w_1 \approx s_1, \ldots, w_m \approx s_m\} \models_\forall X \approx t$, where all the w_i's are constants.

This is actually just a reformulation of the original problem, and one readily proves that S has a unifier if and only if this system of rigid equations is solvable [38, Lemma 5].

Recently it was claimed that second-order unification is undecidabile already when restricted to terms such that all arguments of second-order variables are variable free [130].[4] In the case of s above, this means that all the s_i's must be variable free. If this claim is correct then the undecidability of SREU with ground left-hand sides follows already from the above reduction.

Reduction of Hilbert's tenth problem

In the year 1900 David Hilbert presented a list of 23 problems at a mathematics conference in Paris. The tenth problem was to investigate whether there is a general method for deciding if a diophantine equation has an integer solution or not. A diophantine equation is an equation $p(x_1, \ldots, x_n) = 0$ where $p(\vec{x})$ is a polynomial in variables $\vec{x}$ with coefficients that are integers, e.g., $3x^3y^4 - 5xz + 3 = 0$ is a diophantine equation. The problem was proved undecidable by Matiyasevich 70 years later [99].

As the third undecidability proof of SREU [37], Degtyarev and Voronkov showed how to reduce Hilbert's tenth problem to SREU. The proof is quite short and the key argument [37, Lemma 6] lies in representing multiplication with a system of rigid equations. The idea is to represent multiplication of two positive integers k and l as a list D of pairs, such that the first pair in D is $(0, 0)$ the next $(0 + k, 0 + 1)$, the one after that $(0 + k + k, 0 + 1 + 1)$ and so on. The conditions on D are:

[4] The proof of this claim [130] is very complicated and we have not checked all the details.

1. The first pair of D is $(0, 0)$.
2. For any two consequtive pairs (m, n) and (m', n') in D, $m' = m + k$ and $n' = n + 1$.
3. The last pair of D has the form (x, l) for some x.

These conditions can be expressed by a system of rigid equations with two lists of pairs (in the same spirit as shifted pairing). It follows that $x = kl$. Goldfarb uses the same idea in his proof of the undecidability of second-order unification [64].

Reduction of PCP

The Post Correspondence Problem or PCP over an alphabet Σ can be stated as follows. Given $(v_1, v_2, \ldots, v_k)$ and $(w_1, w_2, \ldots, w_k)$ as two sequences of strings over Σ, is there a sequence $i_1, i_2, \ldots, i_m$, $m \geq 1$, such that

$$w_{i_1} w_{i_2} \cdots w_{i_m} = v_{i_1} v_{i_2} \cdots v_{i_m}$$

This is an undecidable problem [117]. A reduction from PCP to SREU is given in Plaisted [116], where the shifted pairing technique is introduced that we have used in our proof. The important implication is that SREU is undecidable already with ground left-hand sides.

THE HERBRAND SKELETON PROBLEM

4.1 INTRODUCTION

One popular form of the classical Herbrand theorem [74] is this:

> *An existential formula $\exists \vec{x} \varphi(\vec{x})$ is provable if and only if there exist a positive integer m and ground substitutions $\theta_1, \ldots, \theta_m$ in the language of φ such that the disjunction $\varphi\theta_1 \vee \cdots \vee \varphi\theta_m$ is provable.*

The number m is called the *multiplicity*. Multiplicity one may not always suffice. The following example was suggested by Erik Palmgren in a similar context:

$$\varphi(x) \quad = \quad (c \approx 0 \Rightarrow x \approx 1) \wedge (c \approx 1 \Rightarrow x \approx 0)$$

Clearly, $\varphi(0) \vee \varphi(1)$ is provable, but none of the formulas $\varphi(0)$, $\varphi(1)$ or $\varphi(c)$ is provable. An even simpler example is

$$\psi(x) \quad = \quad P(0) \vee P(1) \Rightarrow P(x)$$

where 0 and 1 are constants. Clearly $\psi(0) \vee \psi(1)$ is provable but neither $\psi(0)$ nor $\psi(1)$ is provable. The Herbrand theorem suggests the following approach to automated theorem proving. Given a formula $\varphi(\vec{x})$, first guess the multiplicity m, and then find the appropriate tuples of terms $\vec{t}_1, \ldots, \vec{t}_m$. This gives rise to the following decision problem. Assume that there is at least one constant in the language.

1. *The Herbrand Skeleton Problem*: Given a quantifier free formula φ and a positive integer m, do there exist ground substitutions $\theta_1, \ldots, \theta_m$ such that $\varphi\theta_1 \vee \cdots \vee \varphi\theta_m$ is valid?

The Herbrand Skeleton problem is effectively equivalent to any of the following six decision problems [30].

2. *Formula Instantiation*: the Herbrand skeleton problem with multiplicity one.
3. *Matrix Instantiation*: Given a matrix, is there a substitution that makes every vertical path through that matrix inconsistent?
4. *Existential Intuitionistic*: Is a given existential formula provable in intuitionistic logic?
5. *Prenex Intuitionistic*: Is a given prenex formula provable in intuitionistic logic?
6. *Skeleton Instantiation*: Given a formula and a proof skeleton, is there a derivation of that formula with the given skeleton?

Formula instantiation can be considered as the basic problem that underlies all the other problems. Obviously, the Herbrand skeleton problem of any given multiplicity m reduces to it simply by creating a disjunction of m copies of the given formula.

In the case of logic *without* equality, all the above problems are decidable and reduce to unification. In the case of logic *with* equality, all the above problems are equivalent to SREU [30]. The undecidability of SREU [32, 34, 37, 38] thus implies that problems (1–7) are all undecidable. Note that SREU had several false decidability proofs [53, 56, 66] before it was proved undecidabile, and the problems (1–7) were believed to be decidable.

The Herbrand *m-Skeleton* problem is the Herbrand Skeleton problem with fixed multiplicity m. Clearly, SREU is a special case of the 1-Skeleton problem. Voda and Komara have proved that, for each multiplicity m, the m-Skeleton problem is undecidable [151]. One important conclusion for automated theorem proving, drawn in [151], is that there is no m for which there exists an effective decision procedure that would tell us whether m substitutions suffice to establish the provability of a given quantifier free formula.

Actually, we had a hard time to understand the proof of Voda and Komara until, finally, we convinced ourselves that they have a proof. We wondered if there is a way to derive their result from the Degtyarev–Voronkov theorem. It turns out that indeed there is such a way.

The main result of this chapter is that we show that SREU can be reduced to m-Skeleton problem for any fixed m without adding new nonconstant function symbols. As a corollary, we get a considerably shorter proof of the

undecidability of the m-Skeleton problem. By using results proved in Chapter 3, we can identify, for each multiplicity m, the minimal known fragment of classical logic for which the Herbrand skeleton problem of multiplicity m is undecidable. Our main tool is a logical theorem that we prove first: the *Partisan Corroboration Theorem*. We believe that this theorem is of independent interest.

At the end of this chapter we consider briefly an intriguing generalization of the Herbrand Skeleton problem, suggested recently by Voronkov [157].

4.2 PRELIMINARIES

Atomic formulas and negated atomic formulas are called *positive* and *negative literals* respectively. A *clause* is a disjunction of literals. By a *Horn clause* we mean a clause with exactly one positive literal.[1] A Horn clause is written as $E \Rightarrow s \approx t$ where E is a conjunction of equations, and s and t are terms. By a *Horn formula* we understand a conjunction of Horn clauses.

If $\mathfrak{A}$ is a Σ-structure and $\Sigma' \subseteq \Sigma$ then $\mathfrak{A}\restriction\Sigma'$ is the Σ'-structure that is the reduction of $\mathfrak{A}$ to signature Σ'. Let $\mathfrak{A}$ and $\mathfrak{B}$ be Σ-structures, $\mathfrak{A}$ is a *substructure* of $\mathfrak{B}$, in symbols $\mathfrak{A} \subseteq \mathfrak{B}$, if $A \subseteq B$ and for each n-ary $F \in \Sigma$, $F^{\mathfrak{A}} = F^{\mathfrak{B}}\restriction A^n$.

One easily establishes, by induction on terms and formulas, that if $\mathfrak{A}$ is a substructure of $\mathfrak{B}$ then for all quantifier free sentences φ, $\mathfrak{A} \models \varphi$ iff $\mathfrak{B} \models \varphi$.

Recall that, for any set E of ground equations and for all ground terms s and t,

$$\mathcal{T}_{/E} \models s \approx t \quad \Leftrightarrow \quad E \models s \approx t,$$

where $\mathcal{T}_{/E}$ is the canonical model of E. Recall also that Birkhoff's completeness theorem for equational logic [12] states the following in the case of ground equations. Let E be a set of ground equations and $s \approx t$ a ground equation, then $E \models s \approx t$ iff s can be reduced to t by using the equations in E as rewrite rules in both directions.

4.3 SOME LOGICAL TOOLS

In this section we prove some logical properties that are used in the next section. The main result is Theorem 4.1. The following proposition is actually a consequence of Łoś-Tarski theorem.[2] We say that two (sets of) expressions X and Y are *constant-disjoint* if $\mathcal{C}(X) \cap \mathcal{C}(Y) = \emptyset$.

Proposition 4.1 *Let φ_i for $i \in I$, be pairwise constant-disjoint quantifier free sentences. Then $\models \bigvee_{i\in I} \varphi_i$ implies $\models \varphi_i$ for some $i \in I$.*

[1] By a Horn clause we mean thus a *strict* Horn clause.
[2] Existential sentences are preserved under extensions.

Proof. For $i \in I$, let $\Sigma_i = \Sigma(\varphi_i)$ and let $\Sigma = \bigcup_i \Sigma_i$. Assume that $\bigvee_{i\in I} \varphi_i$ is valid and suppose (by contradiction) that $\not\models \varphi_i$ for all $i \in I$. Then there is (for each $i \in I$) a Σ_i-structure $\mathfrak{A}_i$ such that $\mathfrak{A}_i \models \neg\varphi_i$. Without loss of generality, take all the A_i to be pairwise disjoint.

We now construct a Σ-structure $\mathfrak{A}$ such that $\mathfrak{A}_i \subseteq \mathfrak{A}\restriction\Sigma_i$ for $i \in I$. First let $A = \bigcup_{i\in I} A_i$. For each $i \in I$ and constant $c \in L_i$ let $c^{\mathfrak{A}} = c^{\mathfrak{A}_i}$. For each n-ary function symbol f in Σ define $f^{\mathfrak{A}}$ as follows. For all $\vec{a} = a_1, \ldots, a_n \in A$,

$$f^{\mathfrak{A}}(\vec{a}) = \begin{cases} f^{\mathfrak{A}_i}(\vec{a}), & \text{if } \vec{a} \in A_i; \\ a_1, & \text{otherwise.} \end{cases}$$

It is clear that $\mathfrak{A}$ is well-defined because of the disjointness criteria and that $\mathfrak{A}_i \subseteq \mathfrak{A}\restriction\Sigma_i$ for $i \in I$. Hence $\mathfrak{A}\restriction\Sigma_i \models \neg\varphi_i$, and thus $\mathfrak{A} \models \neg\varphi_i$ for each $i \in I$. But this contradicts that $\models \bigvee_{i\in I} \varphi_i$. ⊠

If we drop the constant-disjointness criterion in Proposition 4.1, then of course the proposition is false. A simple counterexample is

$$\models 0 \approx 1 \vee \neg(0 \approx 1).$$

We state now some other obvious but useful propositions. Proposition 4.2 is an easy corollary of Birkhoff's completeness theorem.

Proposition 4.2 *Let t and s be ground terms and let E and E' be sets of ground equations such that $\mathcal{C}(E') \cap \mathcal{C}(E, s) = \emptyset$, then:*

1. *If $E' \cup E \models t \approx s$ then $E \models t \approx s$.*
2. *If $E \models t \approx s$ then $\Sigma(t) \subseteq \Sigma(E, s)$.*

Proof. Assume that $E' \cup E \models t \approx s$. By Birkhoff's completeness theorem we know that s can be rewritten to t by using $E' \cup E$ as a set of rewrite rules. So there is a sequence of terms $s_0, s_1, \ldots, s_{n-1}, s_n$ where $s_0 = s$, $s_n = t$ and s_i is rewritten to s_{i+1} by using some rule in $E' \cup E$, for $0 \leq i < n$. By induction on i (for $i \leq n$) follows that $\Sigma(s_i) \subseteq \Sigma(E, s)$ and only a rule from E can be used to rewrite s_i. Part 1 follows by Birkhoff's completeness theorem and part 2 follows immediately (take $E' = \emptyset$). ⊠

For a finite set E of equations we write E also for the corresponding conjunction of equations and let the context determine whether a set or a formula is meant.

Proposition 4.3 *Let t and s be ground terms and E' and E sets of ground equations such that E is finite and $\mathcal{C}(E') \cap \mathcal{C}(E, s) = \emptyset$. Then*

$$\mathcal{T}_{/E'\cup E} \models (E \Rightarrow t \approx s) \quad \Rightarrow \quad \models (E \Rightarrow t \approx s).$$

Proof. From $\mathcal{T}_{/E' \cup E} \models (E \Rightarrow t \approx s)$ follows immediately that $\mathcal{T}_{/E' \cup E} \models t \approx s$ and thus $E' \cup E \models t \approx s$. Hence $E \models t \approx s$ by Proposition 4.2, i.e., $\models (E \Rightarrow t \approx s)$. ⊠

We use the following definitions. Let φ be a quantifier free formula and m a positive integer.

- A set of m ground substitutions Θ is an *m-corroborator* for φ if

$$\models \bigvee_{\theta \in \Theta} \varphi\theta.$$

  When $\Theta = \{\theta\}$ consists of a single substitution θ, then we say that θ is a *corroborator* for φ or *corroborates* φ.

So the *m-Skeleton problem* is the problem of existence of m-corroborators for given formulas.

- For $x \in \mathcal{V}(\varphi)$, a *guard for x in φ*, if it exists, is a clause

$$E \Rightarrow t \approx s$$

  in φ such that E and s are ground and x occurs in t. We say that

$$\bigwedge_{x \in \mathcal{V}(\varphi)} \psi_x$$

  is a *guard* of φ if each ψ_x is a guard for x in φ; φ is is called *guarded* if it has a guard.

Intuitively, in the light of the second part of Proposition 4.2, the notion of a Horn formula being guarded is a sufficient condition to guarantee that if there is a corroborator θ for φ then the range of $\theta \restriction \mathcal{V}(\varphi)$ is $\mathcal{T}_{\Sigma(\varphi)}$, i.e., $\Sigma(\varphi\theta) = \Sigma(\varphi)$.

SREU is, by definition, the problem of existence of corroborators for Horn formulas. However, we only need to consider *guarded* Horn formulas. To see that, consider a Horn formula φ; let Σ be its signature expanded with a constant if φ has no constants and let c be a constant in Σ. Let $\varphi'(x)$ be the Horn clause $E_\Sigma \Rightarrow x \approx c$ where[3]

$$E_\Sigma = \{\, f(c, \ldots, c) \approx c \mid f \in \Sigma \,\}.$$

Let now ψ be the guarded Horn formula

$$(\bigwedge_{x \in \mathcal{V}(\varphi)} \varphi'(x)) \wedge \varphi.$$

[3]Note that when f is a constant then $f(c, \ldots, c)$ stands for f.

Clearly, ψ has a corroborator iff φ has one. Note that, for all terms t,

$$\models (E_\Sigma \Rightarrow t \approx c) \quad \Leftrightarrow \quad t \in \mathcal{T}_\Sigma.$$

Example 4.1 A simple example of a guarded Horn formula is this:

$$\begin{aligned} \psi \quad = \quad & (E_1 \Rightarrow c_1' \,.\, x \approx c_1) \wedge \\ & (E_2 \Rightarrow c_2' \,.\, y \approx c_2) \wedge \\ & (\Pi_1 \Rightarrow x \approx y) \wedge \\ & (\Pi_2 \Rightarrow x \approx t \,.\, y), \end{aligned}$$

where E_1, E_2, Π_1, Π_2 and t are ground, c_1, c_1', c_2 and c_2' are constants and $.$ is a binary function symbol. The guard of ψ is

$$(E_1 \Rightarrow c_1' \,.\, x \approx c_1) \wedge (E_2 \Rightarrow c_2' \,.\, y \approx c_2).$$

An example of a Horn formula with a common guard for all variables is

$$\begin{aligned} \varphi \quad = \quad & (E \Rightarrow x \,.\, y \approx c) \wedge \\ & (\Pi_1 \Rightarrow x \approx y) \wedge \\ & (\Pi_2 \Rightarrow x \approx t \,.\, y), \end{aligned}$$

where E, Π_1, Π_2 and t are ground and c is a constant. The guard of φ is

$$E \Rightarrow x \,.\, y \approx c.$$

Note that the above formulas have the same structure as the systems of rigid equations S_v^M and $\hat{S}_v^M$ in Chapter 3. □

We use the following definition.

- A corroborator of a disjunction φ is *partisan*, if it corroborates some disjunct of φ.

The main result of this section is the following theorem.

Theorem 4.1 (Partisan Corroboration Theorem) *Any corroborator of a disjunction of constant-disjoint guarded Horn formulas is partisan.*

Proof. Let $\varphi = \bigvee_{i \in I} \varphi_i$ where all the φ_i's are constant-disjoint guarded Horn formulas. Let θ be a corroborator for φ. We must prove that θ corroborates φ_i for some $i \in I$.

We can assume (without loss of generality) that there exist positive integers m and n such that each φ_i has the following form:

$$\varphi_i \quad = \quad \underbrace{\bigwedge_{1\leq k\leq m} (E_i^k \Rightarrow s_i^k \approx t_i^k)}_{\psi_i} \quad \wedge \quad \bigwedge_{1\leq k\leq n} (D_i^k \Rightarrow u_i^k \approx v_i^k),$$

where ψ_i is a guard of φ_i, i.e., each E_i^k and s_i^k is ground and $\mathcal{V}(\varphi_i) = \mathcal{V}(\psi_i)$, for all $i \in I$. Let $C_i = \mathcal{C}(\varphi_i)$ for $i \in I$. We have that

$$C_i \cap C_j = \emptyset \quad (\forall i, j \in I,\ i \neq j). \tag{4.1}$$

Let $\Sigma = \Sigma(\varphi)$. For $i \in I$ let $\mathcal{K}_i$ denote the class of all Σ-structures that satisfy $\varphi_i\theta$, i.e,

$$\mathcal{K}_i = \{\, \Sigma\text{-structure } \mathfrak{A} \mid \mathfrak{A} \models \varphi_i\theta \,\}.$$

From the validity of $\varphi\theta$ follows that each Σ-structure belongs to some $\mathcal{K}_i$.

Let now J be any subset of I such that

$$\models \psi_i\theta \quad (\forall i \in J). \tag{4.2}$$

(Take for example $J = \emptyset$.) So

$$\mathcal{C}(\varphi_i\theta) = C_i \quad (\forall i \in J). \tag{4.3}$$

To see that (4.3) holds, suppose (by contradiction) that $\mathcal{C}(\varphi_i\theta)$ contains some $c \notin C_i$. Clearly, c belongs to some $x\theta$ where x occurs in the guard ψ_i. By the second part of Proposition 4.2, every constant in $x\theta$ belongs to C_i. This gives the desired contradiction.

If $I = J$ then the theorem follows by Proposition 4.1. Assume that $I \neq J$. Now we prove the following statement:

$$\text{If } \not\models \varphi_i\theta \text{ for all } i \in J \text{ then } \models \psi_i\theta \text{ for some } i \in I \setminus J. \tag{4.4}$$

Proof of (4.4) Assume $\not\models \varphi_i\theta$ for all $i \in J$. Form an equation set D as follows.

- If $J = \emptyset$ let $D = \emptyset$.
- If $J \neq \emptyset$ then there is for each $i \in J$ a clause in $\varphi_i\theta$ that is not valid and by (4.2) this clause is not in $\psi_i\theta$. In other words, there is a mapping $f : J \to \{1, 2, \ldots, n\}$ such that

  $$\not\models (D_i^{f(i)} \Rightarrow u_i^{f(i)} \approx v_i^{f(i)})\theta \quad (\forall i \in J). \tag{4.5}$$

  Let f be fixed and let $D = \bigcup_{i\in J} D_i^{f(i)}\theta$.

For each mapping $g : I \setminus J \to \{1, 2, \ldots, m\}$ let E_g denote the following set of equations:

$$E_g = \bigcup_{i \in I \setminus J} E_i^{g(i)},$$

and let $\mathfrak{A}_g$ be the canonical model of $D \cup E_g$, i.e.,

$$\mathfrak{A}_g = \mathcal{T}_{/E_g \cup D}.$$

We can prove now the following statement.

(*) Fix $g : I \setminus J \to \{1, 2, \ldots, m\}$. There exists an $i \in I \setminus J$ such that $\mathfrak{A}_g \in \mathcal{K}_i$.

Proof of (*) Assume that (*) doesn't hold. (Assume also that $J \neq \emptyset$ or else (*) holds trivially.) Then $\mathfrak{A}_g \in \mathcal{K}_j$ for some $j \in J$. Fix such an appropriate j.

So $\mathfrak{A}_g$ satisfies each clause in $\varphi_j\theta$ and in particular

$$\mathfrak{A}_g \models (D_j^{f(j)} \Rightarrow u_j^{f(j)} \approx v_j^{f(j)})\theta.$$

Let $D' = D_j^{f(j)}\theta$, $u' = u_j^{f(j)}\theta$ and $v' = v_j^{f(j)}\theta$. By (4.3) follows that

$$\mathcal{C}(D', u', v') \subseteq C_j$$

and

$$\begin{aligned} \mathcal{C}(E_g, D \setminus D') &= \mathcal{C}(E_g) \cup \mathcal{C}(D \setminus D') \\ &= \mathcal{C}(E_g) \cup \bigcup_{i \in J, i \neq j} \mathcal{C}(D_i^{f(i)}\theta) \\ &\subseteq \bigcup_{i \in I \setminus J} C_i \cup \bigcup_{i \in J, i \neq j} C_i \\ &= \bigcup_{i \in I, i \neq j} C_i. \end{aligned}$$

So, by (4.1),

$$\mathcal{C}(D', u', v') \cap \mathcal{C}(E_g, D \setminus D') = \emptyset.$$

It follows, by Proposition 4.3, that

$$\models (D_j^{f(j)} \Rightarrow u_j^{f(j)} \approx v_j^{f(j)})\theta.$$

But this contradicts (4.5).

By using (*) we can now prove the following statement

(**) There exists an $i \in I \setminus J$ such that $\models \psi_i\theta$.

Proof of (**) Assume that the claim is wrong.

Then there is for each $i \in I \setminus J$ a clause in $\psi_i\theta$ that is not valid, i.e., there is a mapping $g : I \setminus J \to \{1, 2, \ldots, m\}$ such that

$$\not\models E_i^{g(i)} \Rightarrow s_i^{g(i)} \approx (t_i^{g(i)}\theta) \quad (\forall i \in I \setminus J).$$

(Note that only the t_i's can be nonground.) Fix such an appropriate g.

By using (*) we know that $\mathfrak{A}_g \in \mathcal{K}_i$ for some $i \in I \setminus J$. Choose such an i. So $\mathfrak{A}_g$ satisfies each clause in $\varphi_i\theta$ and in particular

$$\mathfrak{A}_g \models E_i^{g(i)} \Rightarrow s_i^{g(i)} \approx (t_i^{g(i)}\theta).$$

But, by (4.3) and (4.1), $\mathcal{C}(E_i^{g(i)}, s_i^{g(i)}) \cap \mathcal{C}(E_g \setminus E_i^{g(i)}, D) = \emptyset$. Hence, by Proposition 4.3,

$$\models E_i^{g(i)} \Rightarrow s_i^{g(i)} \approx (t_i^{g(i)}\theta).$$

So we have contradiction.

This proves statement (4.4). Let now J be the *maximal* subset of I such that (4.2) holds. In other words, for all $i \in I \setminus J$, $\not\models \psi_i\theta$. By the contrapositive of (4.4) we conclude that for some $i \in J$, $\models \varphi_i\theta$ and the theorem follows. ⊠

Remark Theorem 4.1, as well as its proof, remain correct if the disjunction is infinite. We do not use this generalization.

The following example illustrates why the conditions of being constant-disjoint and guarded are important and cannot in general be discarded. In each case there is a counterexample to the theorem.

Example 4.2 Let us first consider an example where the disjuncts are guarded but not constant-disjoint. Let $\varphi(x)$ be the following guarded Horn formula:

$$(c \approx 0 \Rightarrow x \approx 1) \wedge (c \approx 1 \Rightarrow x \approx 0)$$

where c, 0 and 1 are contants, and let $\varphi_1 = \varphi(x_1)$, $\varphi_0 = \varphi(x_0)$ and $\psi = \varphi_1 \vee \varphi_0$ where x_1 and x_0 are distinct variables. Consider now any ground

substitution θ such that $\theta(x_1) = 1$ and $\theta(x_0) = 0$. It is easy to show by case analysis that θ corroborates ψ, i.e., that

$$\begin{aligned} \models \quad & ((c \approx 0 \Rightarrow 1 \approx 1) \wedge (c \approx 1 \Rightarrow 1 \approx 0)) \vee \\ & ((c \approx 0 \Rightarrow 0 \approx 1) \wedge (c \approx 1 \Rightarrow 0 \approx 0)). \end{aligned}$$

However, θ corroborates neither φ_1 nor φ_0.

Let us now consider the case when constant-disjointness is not violated but the disjuncts are not guarded. Let $\varphi_1(y, x_1, y_1)$ be the formula

$$((y \approx 0 \Rightarrow x_1 \approx y_1) \wedge (y \approx y_1 \Rightarrow x_1 \approx 0))$$

and let $\varphi_0(x_0, y_0)$ be the formula

$$((c \approx y_0 \Rightarrow x_0 \approx 1) \wedge (c \approx 1 \Rightarrow x_0 \approx y_0))$$

where c, 0 and 1 are constants and x_1, x_0, y_1, y_0, y distinct variables. Let $\psi = \varphi_1 \vee \varphi_0$. Let θ be a ground substitution such that $\theta(x_1) = 1$, $\theta(x_0) = 0$, $\theta(y) = c$, $\theta(y_1) = 1$ and $\theta(y_0) = 0$. Then $\models \psi\theta$ but $\not\models \varphi_1\theta$ and $\not\models \varphi_0\theta$ (the situation is exaclty the same as in the previous case). □

4.4 FROM 1-SKELETON TO N-SKELETON PROBLEM

The 1-Skeleton problem is undecidable. This follows from the undecidability of SREU by Degtyarev and Voronkov [34, 38]. We can formulate their result in the current setting as follows (cf [38, Theorem 1]).

Theorem 4.2 (Degtyarev–Voronkov) *The 1-Skeleton problem of guarded Horn formulas is undecidable.*

Under certain restrictions on the language and the structure of formulas, the 1-Skeleton problem becomes decidable. As we have shown in Chapter 3, 1-Skeleton problem is already undecidable in the presence of one binary function symbol (in addition to constants); moreover, two variables suffice for undecidability. In the case one variable the problem becomes decidable as is shown in Chapter 6.

For technical reasons it is convenient to assume that we have a fixed signature Σ with $\{c_1, c_2, \ldots\}$ as the set of distinct constants in it. Σ may also have other function symbols of arity ≥ 1. Let us also be precise about the variables that we allow in Σ-expressions, by assuming that all variables come from the collection $\{x_1, x_2, \ldots\}$.

For each natural number n, constant c and variable x, let $c^{(n)}$ denote a new constant and let $x^{(n)}$ denote a new variable. We define by induction on

any Σ-expression X the corresponding expression $X^{(n)}$ as the one obtained from X by replacing in it each variable x with $x^{(n)}$ and each constant c with $c^{(n)}$. For any substitution θ of Σ-variables with Σ-terms we let $\theta^{(n)}$ denote a substitution that takes the variable $x^{(n)}$ to the term $\theta(x)^{(n)}$. So, for any Σ-expression X and natural number n,

$$(X\theta)^{(n)} = X^{(n)}\theta^{(n)}.$$

The following property is immediate. For any Σ-sentence φ and natural number n,

$$\models \varphi \quad \Leftrightarrow \quad \models \varphi^{(n)}.$$

Theorem 4.3 *Let φ be a guarded Horn formula and n a positive integer. Then φ has a corroborator iff $\bigwedge_{i=1}^{n} \varphi^{(i)}$ has an n-corroborator.*

Proof. The '$\Rightarrow$' direction is trivial. We prove the '$\Leftarrow$' direction as follows. Let $I = \{1, 2, \ldots, n\}$ and let ψ be the formula $\bigwedge_{i\in I} \varphi^{(i)}$. Assume that ψ has an n-corroborator $\{\, \theta_i \mid i \in I \,\}$. So

$$\models \bigvee_{i\in I} (\bigwedge_{j\in I} \varphi^{(j)}\theta_i).$$

By the distributive law this is equivalent to

$$\models \bigwedge_{f:I\to I} (\bigvee_{i\in I} \varphi^{(f(i))}\theta_i).$$

From this follows in particular that

$$\models \bigvee_{i\in I} \varphi^{(i)}\theta_i.$$

Let $X_i = \mathcal{V}(\varphi^{(i)})$ for $i \in I$. Since all the X_i's are pairwise disjoint we can let θ' be a substitution such that $\theta' \restriction X_i = \theta_i \restriction X_i$ for $i \in I$, and it follows that

$$\models \bigvee_{i\in I} \varphi^{(i)}\theta'.$$

By Theorem 4.1 follows now that $\models \varphi^{(i)}\theta'$ for some $i \in I$. Fix such an appropriate i. But then, by Proposition 4.2, the range of $\theta' \restriction X_i$ is $\mathcal{T}_{\Sigma(\varphi^{(i)})}$, and thus there is a substitution θ with range $\mathcal{T}_\Sigma$ such that $\theta^{(i)} \restriction X_i = \theta' \restriction X_i$. Hence $\models \varphi^{(i)}\theta^{(i)}$ and so $\models \varphi\theta$ by above. ⊠

Corollary 4.1 (Voda–Komara) *For all $n \geq 1$, n-Skeleton problem is undecidable.*

Proof. The reduction in Theorem 4.3 is trivially effective. So, if we had a decision procedure (for some n) for finding n-corroborators, we could use it to find corroborators, but this would contradict Theorem 4.2. ⊠

Assume that we are using an automated theorem proving method that is based on the Herbrand theorem. Roughly, this involves a search for terms, for a given bound m on multiplicity. Corollary 4.1 (Voda and Komara [151]) tells us that there is no m for which we could effectively decide when to stop our search for such terms in case they don't exist.

Recall that *monadic SREU* is SREU restricted to signatures with function symbols of arity ≤ 1. The decidability of monadic SREU is currently one of the few open problems related to SREU [73]. An effectively equivalent problem is the decidability of the prenex fragment of intuitionistic logic with equality with function symbols of arity ≤ 1 [35]. Some evidence speaks in favour of that the problem is decidable, although with very high computational complexity (e.g., many subcases are decidable). We get also the following result.

Corollary 4.2 *If the n-Skeleton problem is decidable in the monadic case, for some $n > 1$ then so is monadic SREU.*

Proof. Note that, given a formula φ, the formula $\bigwedge_{i=1}^{n} \varphi^{(i)}$ contains the same nonconstant function symbols as φ. The rest follows by Theorem 4.3. ⊠

One might show decidability of monadic SREU by keeping Corollary 4.2 in mind and first show that the monadic n-Skeleton problem is decidable for some $n > 1$. This may be easier than a direct proof, due to the freedom of choice of n.

Minimal Case

Consider the system of rigid equations, constructed in Chapter 3:

$$\hat{S}^{\mathrm{u}}(z, x, y) = \{E \models_{\forall} x \centerdot y \approx q,\ \Pi_1 \models_{\forall} x \approx y,\ \Pi_2 \models_{\forall} x \approx z \centerdot y\}.$$

Recall that E, Π_1 and Π_2 are gound and, for any TM M and input string v for M, the system $\hat{S}^{\mathrm{u}}(t_{\langle M,v\rangle}, x, y)$ is solvable iff $\hat{S}^{M}_{v}$ is solvable iff M accepts v, where the term $t_{\langle M,v\rangle}$ represents the encoding of the pair (M, v). Let $\varphi^{\mathrm{u}}(z, x, y)$ be the corresponding formula:

$$\varphi^{\mathrm{u}}(z, x, y) = (E \Rightarrow x \centerdot y \approx q) \wedge (\Pi_1 \Rightarrow x \approx y) \wedge (\Pi_2 \Rightarrow x \approx z \centerdot y).$$

So, for all substitutions θ, θ corroborates $\varphi^{\mathrm{u}}(t_{\langle M,v\rangle}, x, y)$ iff M accepts v. We get the following result.

Corollary 4.3 *For all $n \geq 1$, the n-Skeleton problem of guarded Horn formulas restricted to $2n$ variables and $3n$ clauses with ground negative literals is undecidable, already for some fixed negative literals*

Proof. Let M be a TM and v an input string for M, and let n be a positive integer. So

$$\psi \quad = \quad \bigwedge_{i=1}^{n} (\varphi^{\mathrm{u}}(t_{\langle M,v\rangle}, x, y))^{(i)}$$

is a guarded Horn formula with $2n$ variables and $3n$ clauses. Furthermore, the negative literals in ψ are fixed for any fixed n. The statement follows by Theorem 4.3 and Theorem 3.3. ⊠

4.5 HERBRAND F-SKELETON PROBLEM

The automated theorem proving methods that are based on the Herbrand theorem are in general called *rigid variable methods* [157]. The principal procedure for such methods can be described as follows. Let $\varphi(\vec{x})$ be a quantifier free formula.

Step I Choose a multiplicity m.

Step II Check if $\varphi(\vec{x})$ has an m-corroborator.

Step III If an m-corroborator exists then $\exists\vec{x}\varphi(\vec{x})$ is valid, otherwise increase m and go to Step II.

Voronkov investigates the complexity of various problems related to such methods [157]. In particular, he considers the rigid-variable methods in the context of a fragment of classical logic for which validity is *decidable*, and proves that, for this fragment, a rigid-variable method (by Gallier et al [52, 53, 50]) introduces (by using Plaisteds result [116]) an *undecidable* sub-problem at Step II. He notes that the result of Voda and Komara [151] simply shows the inadequacy of the formulation of the Herbrand Skeleton problem and suggests the notion of *strategy for multiplicity*.

Strategies for Multiplicity

Informally, a strategy for multiplicity is a procedure that selects the initial multiplicity for Step I and then increases the multiplicity each time Step II is re-entered. The *standard* strategy is the one that, initially, chooses $m = 1$ and then increments m by one each time Step II is rerun. A strategy is called *formula-independent* if it does not depend on φ. The formal definition is as follows.

- *A strategy (for multiplicity)*, is a function f whose first argument is a quantifier free formula, second argument is a natural number and the range of f is the set of natural numbers.

The second argument is the number of times Step II has been re-executed. We say that a strategy f is *increasing* if f satisfies the following property, for all $k, l \in \mathbb{N}$,

$$k < l \quad \Rightarrow \quad f(\varphi, k) < f(\varphi, l).$$

So, an inreasing strategy is such that, each time Step II is re-executed, the multiplicity is increased. Let f be a strategy. The following decision problem arises at Step II:

- *The f-Skeleton Problem.* Given a quantifier free formula φ and $k \in \mathbb{N}$, does φ have an $f(\varphi, k)$-corroborator?

Unless otherwise stated, by strategy we mean *computable* strategy. Clearly, existence of a *noncomputable* (increasing) strategy follows by the Herbrand theorem. Voronkov poses the following problem:

> Does there exist an increasing strategy f for which the f-Skeleton problem is decidable?

This problem is currently open.

Some Special Cases

For some classes of formulas, a strategy for multiplicity can be shown to exist. Consider the following class of formulas [157]. A variable x is said to occur *positively* in a formula φ if x has an occurrence in φ which is within the scope of an even number of negation symbols.[4]

- A closed formula φ is *ground-negative* if all variables in it occur positively.[5]

It is pointed out in [157] that the validity problem for ground-negative formulas is decidable. However, the systems of rigid equations that arise from such formulas are with ground left-hand sides and thus their solvability undecidable by Plaisted's result [116]. It is also shown that for ground-negative formulas there exists a (nonincreasing) strategy, basically, a function that given a ground-negative formula φ (the second argument is not used) returns the multiplicity m such that φ is valid iff φ has an m-corroborator. The following result is shown in [157], by using Corollary 4.3.

[4] Here the formula φ is assumed to contain only the connectives '$\wedge$', '$\vee$' and '$\neg$', to avoid "hidden" negation symbols.

[5] The definition in [157] is slightly more general.

Theorem 4.4 (Voronkov) *For any formula-independent strategy for the class of ground-negative formulas, Step II is undecidable.*

Further Remarks

Note that the validity problem of existential closures of Horn formulas with ground negative literals, like $\exists x \exists y \varphi^{\mathrm{u}}(t_{\langle M,v\rangle}, x, y)$ for example, is *decidable*, since such formulas are ground-negative. At the same time the 1-Skeleton problem of $\varphi^{\mathrm{u}}(t_{\langle M,v\rangle}, x, y)$ is *undecidable* as we have shown.

We can also note the following "reversed" fenomenon. The following problem is *undecidable* if there is either one binary or two unary function symbols in the signature [159, 160] and no relation symbols besides equality (cf [13, Corollary 4.1.3]):

- Given a Horn clause $\varphi(\vec{x})$, is $\exists \vec{x} \varphi(\vec{x})$ valid?

Actually, it is enough that there are three literals and three variables in $\varphi(\vec{x})$ [160]. On the other hand, we know that the 1-Skeleton problem of Horn clauses is decidable, this is just rigid E-unification.

See also Chapter 7 (Section 7.5).

FINITE TREE AUTOMATA

5.1 INTRODUCTION

Finite tree automata [43, 143] is a natural generalization of classical finite automata to automata that accept or recognize trees of symbols, not just strings. In the deterministic case, this generalization is best understood by first looking at a deterministic finite automaton with input alphabet Σ as a finite $(\{\epsilon\} \cup \Sigma)$-structure with the elements of its universe as states, where ϵ is a constant and the symbols in Σ are unary function symbols.[1] The generalization consists of arbitrary (not just unary) function symbols in Σ. The recognizability condition of a ground (or closed) term is, like in the unary case, simply that its value is a final state.

Many decision problems concerned with finite automata (non-emptiness, inequivalence, etc.) have natural counterparts with finite tree automata. As in the case of finite automata, decision problems of finite tree automata are typically complete for the computational complexity classes they belong to and, due to their simple formulation, have proved to be useful tools in classifying complexity bounds of other problems. In particular, inequivalence [127, 132] and intersection non-emptiness [29, 48, 133] are examples of such decision problems.

The intersection non-emptiness problem of finite tree automata arises naturally in the context of type inference in logic programming [48]. The same decision problem restricted to top-down deterministic finite tree automata arises also in sort inference in typed functional programming [133]. Our main motivation for studying this problem is its close connection with SREU. These connections are investigated in Chapter 6.

[1] The idea is that the interpretation of ϵ is the initial state and that the interpretation of a unary function symbol σ is a function $\boldsymbol{\sigma}$ such that there is a transition with label σ from a state q to a state p iff $\boldsymbol{\sigma}(q) = p$. So the value of a term $\sigma_1(\sigma_2(\cdots\sigma_n(\epsilon)))$ is the state after reading the string $\sigma_n \cdots \sigma_2\sigma_1$. This observation is attributed to Büchi and Wright [15].

The main contributions of this chapter can be summarized as follows. We present a comprehensive proof of *EXPTIME-completeness of the intersection non-emptiness problem of finite tree automata*. More precisely, it is proved that the hardness result holds already for *deterministic* finite (bottom-up) tree automata. Although the complexity of this problem has been used in the above mentioned contexts and also in the context of a "decidability" proof of SREU [66], its proof is either merely remarked upon [48], or only briefly outlined and incomplete [66, 133]. The proof of its complexity is however highly nontrivial and in order to trust it we had to prove it ourselves. In general, it was very hard to find complexity results related to the basic decision problems of finite tree automata, as they are scattered throughout the literature, and we decided to make a short survey by collecting the complexity results of the closely related problems. This survey is summarized in Table 5.1 at the end of this chapter.

We show also that the *non-emptiness problem of finite tree automata is P-complete* by showing its close connection with the two well-known P-complete problems *alternating graph accessibility* [68, 80] and *generability* [68, 82, 89]. We consider a notion of succinctness with respect to which the intersection non-emptiness problem is in fact a *succinct* version of the non-emptiness problem. We believe that these decision problems of finite tree automata will appear in other contexts and expect that this survey will be useful therein. In general we conclude that there is a rule of thumb:

> If a decision problem for (deterministic) finite automata is complete for a certain space complexity class, then the same decision problem for (deterministic) finite *tree* automata is complete for the corresponding *alternating* space complexity class.

But alternating space is precisely deterministic time, only one exponential higher [17].

5.2 PRELIMINARIES

Finite Tree Automata

Let us recall the definition of a (bottom-up) tree automaton.

- A *tree automaton* or *TA* A is a quadruple (Q, Σ, R, F) where
  - Q is a finite set of *states*,
  - Σ is a *signature* or an *input alphabet* disjoint from Q,
  - R is a set of *rules* of the form $\sigma(q_1, \ldots, q_n) \to q$, where $\sigma \in \Sigma$ has arity $n \geq 0$ and $q, q_1, \ldots, q_n \in Q$,

– $F \subseteq Q$ is the set of *final states*.

A is called a *deterministic* TA or DTA if there are no two different rules in R with the same left-hand side.

Tree automata as defined above are usually also called *bottom-up* tree automata. Top-down tree automata were introduced by Rabin [121] and were also studied by Magidor and Moran [95]. Here we use the following definition, also based on rewrite rules.

▶ A *top-down tree automaton* or *TTA* A is a quadruple (Q, Σ, R, I) where Q and Σ are as above,

– R is a set of *rules* of the form $q \to \sigma(q_1, \ldots, q_n)$, where $\sigma \in \Sigma$ has arity $n \geq 0$ and $q, q_1, \ldots, q_n \in Q$,

– $I \subseteq Q$ is the set of *initial states*.

A is called a *deterministic* TTA or DTTA if I is a *singleton set*, and whenever $q \longrightarrow_R \sigma(\vec{q})$ and $q \longrightarrow_R \sigma(\vec{p})$ then $\vec{q} = \vec{p}$.

Terms are also called trees. A set of terms (or trees) is called a *forest*. Recognizability for tree automata (either bottom-up or top-down) is defined as follows.

▶ The forest *recognized* by a TA $A = (Q, \Sigma, R, X)$ (or a TTA $A = (Q, \Sigma, R^{-1}, X)$) is the set

$$T(A) = \{\, \tau \in \mathcal{T}_\Sigma \mid (\exists q \in X)\ \tau \xrightarrow{*}_R q \,\}.$$

A forest is called *recognizable* if it is recognized by some TA (or TTA).

Recall that two tree automata are *equivalent* if they recognize the same forest. It is well-known that the nondeterministic and the deterministic versions of TAs have the same expressive power [43, 60, 143], i.e., for any TA there is an equivalent DTA. Clearly there is no essential difference between a TA and a TTA. However, the class of forests recognized by DTTAs are properly contained in the class of all recognizable forests. A simple example of this is the forest $\{f(a,b), f(b,a)\}$ that is clearly recognizable but not by any DTTA [60, Example 2.11].

We say that a TA is *total* if every term over its input alphabet reduces to some state. Every TA can trivially be extended (by adding new rules and a new dummy state) to an equivalent total TA. Every total DTA $A = (Q, \Sigma, R, F)$ can be seen as a pair $(\mathfrak{A}, F)$, where $\mathfrak{A}$ is a Σ-structure with

universe Q whose interpretation function is determined by R as follows: for all $f \in \Sigma$ (of arity n) and $q, q_1, \ldots, q_n \in Q$,

$$f^{\mathfrak{A}}(q_1, \ldots, q_n) = q \quad \Leftrightarrow \quad f(q_1, \ldots, q_n) \longrightarrow_R q.$$

Then we have that

$$T(A) = \{\, \tau \in \mathcal{T}_\Sigma \mid \tau^{\mathfrak{A}} \in F \,\}. \tag{5.1}$$

Conversely, any pair $(\mathfrak{A}, F)$ where $\mathfrak{A}$ is a finite Σ-algebra and F a subset of its universe, can be seen as a DTA. This is actually the definition of a DTA used by Gécseg and Steinby [60]. We note that the study of various forms of recognizability is a research area by itself [21, 103].

Alternation and Computational Complexity

Alternation was introduced by Chandra, Kozen and Stockmeyer [17] as a generalization of nondeterminism. First, let us give an intuitive definition of an alternating Turing machine or ATM. An ATM is like a nondeterministic Turing machine (TM), except that every configuration or instantaneous description (ID) is labelled as either "universal" or "existential". Actually, each state is either universal or existential and an ID is labelled accordingly.[2] We inductively determine if an ID "leads to acceptance" as follows. Any final ID leads to acceptance. For any nonfinal ID we have two cases: an existential ID leads to acceptance if at least one of its successors leads to acceptance; a universal ID leads to acceptance if all of its successors lead to acceptance and it has at least one successor.

All computation models based on a Turing machine can be considered as variants of a TM with different acceptance conditions, this point is emphasized by Johnson [81]. We define an ATM formally as follows.

- An *alternating Turing machine* is a pair (M, U) where M is a TM and U a subset of the states of M, called the set of *universal states*. The states of M not in U are called *existential*.

An ATM with an empty set of universal states is simply a TM. An ID of an ATM is said to be *existential* (respectively *universal*, *final*, *initial*) if its state is existential (respectively universal, final, initial). We can now formally define the notion of acceptance for ATMs.

- Let M be an ATM with initial state q_0 and x a string over its input alphabet. Then M *accepts* x iff the initial ID q_0x, leads to acceptance, where *leads to acceptance* is defined recursively as follows.

[2] In the original definition of an ATM there is also a possibility of a "negated" state, but it can be omitted without loss of generality [17, Theorem 2.5].

- Any final ID leads to acceptance.
- If v is a nonfinal ID then it leads to acceptance iff
  - v is existential and some successor of v leads to acceptance, or
  - v is universal, all successors of v lead to acceptance and v has at least one successor.

Note that the acceptance condition of an ATM without universal states is the same as the acceptance condition of the underlying TM.

Alternating Space vs Deterministic Time The notion of space (and time) complexity of ATMs is the same as that of TMs. The key property that we are going to use is that, *alternating space is precisely deterministic time, only one exponential higher* [17]. In particular,

- APSPACE = EXPTIME,
- ALOGSPACE = P,

where the classes APSPACE and ALOGSPACE consist of all problems that can be solved by a polynomial space ATM and a logarithmic space ATM, respectively.

5.3 BASIC DECISION PROBLEMS

All the basic decision problems of finite tree automata, like the *non-emptiness problem*, the *inequivalence problem* (or the more general *inclusion problem*) are decidable (see Gécseg and Steinby [60]). The proofs are fairly easy by first transforming a TA into a DTA by a powerset construction and then using a "pumping property" for DTAs. It is also easy to show that recognizable sets of terms are closed under Boolean operations. This is illustrated next.

- **Complementation:** Let $A = (Q, \Sigma, R, F)$ be a total DTA. The *complement* of A is the DTA $\bar{A} = (Q, \Sigma, R, Q \setminus F)$. It follows immediately from (5.1) that $T(\bar{A}) = \mathcal{T}_\Sigma \setminus T(A)$.
- **Intersection:** Let $A = (Q_1, \Sigma, R_1, F_1)$ and $B = (Q_2, \Sigma, R_2, F_2)$ be TAs. The *direct product* of A and B is the TA

$$A \times B = (Q_1 \times Q_2, \Sigma, R, F_1 \times F_2),$$

where R is the set of rules $f((a_1, b_1), \ldots, (a_n, b_n)) \to (a, b)$ such that $f(\vec{a}) \longrightarrow_{R_1} a$ and $f(\vec{b}) \longrightarrow_{R_2} b$. It follows easily that

$$T(A \times B) = T(A) \cap T(B).$$

Note that if A and B above are total DTAs then so is their direct product. Let A and B be total DTAs. Clearly the inclusion and inequivalence problems for DTAs reduce effectively to the non-emptiness problem, since $T(A) \subseteq T(B)$ iff $T(A) \cap T(\bar{B}) = \emptyset$. It follows for example that

$$\begin{aligned} T(A) = T(B) \quad &\Leftrightarrow \quad (T(A) \cap T(\bar{B})) \cup (T(B) \cap T(\bar{A})) = \emptyset \\ &\Leftrightarrow \quad T(\overline{\overline{A \times \bar{B}} \times \overline{B \times \bar{A}}}) = \emptyset \end{aligned} \tag{5.2}$$

In the following two sections we address the following decision problems.

- *Non-emptiness of TAs* (or, more particularly, of DTAs or DTTAs) is the following decision problem: Given a finite tree automaton A, is $T(A)$ non-empty?

- *Inequivalence of TAs* (or, more particularly, of DTAs or DTTAs) is the following decision problem: Given finite tree automata A and B with the same signature, are $T(A)$ and $T(B)$ unequal?

- *Intersection non-emptiness of TAs* (or, more particularly, of DTAs or DTTAs) is the following decision problem: Given a finite sequence $(A_i)_{i<n}$ of finite tree automata, is $\bigcap_{i<n} T(A_i)$ non-empty?

For finite automata the same decision problems are defined analogously. It is clear that, by using (5.2), inequivalence of DTAs reduces (in logarithmic space) to non-emptiness [60]. For DFAs this was already shown by Moore [105]. It is also clear that for a fixed n, the intersection non-emptiness problem reduces (in logarithmic space) to the non-emptiness problem.

5.4 NON-EMPTINESS AND INEQUIVALENCE

For finite automata the non-emptiness problem is basically the same as the *graph accessibility problem* and is thus complete for nondeterministic logarithmic space or NL-complete [128]. It follows that the inequivalence problem of DFAs is also NL-complete. Analogously, for finite tree automata there is a simple reduction from the *alternating graph accessibility problem* to the non-emptiness problem and vice versa. Alternating graph accessibility was shown P-complete by Immerman [80] by a direct simulation of any ALOGSPACE ATM. There is also a very simple reduction from *generability*, which is another P-complete problem due to Jones and Laaser [82] and Kozen [89], to non-emptiness of DTAs and vice versa. We follow Greenlaw, Hoover and Ruzzo [67, 68] in our formulation of alternating graph accessibility and generability.[3]

[3] The book of Greenlaw, Hoover and Ruzzo [68] includes an excellent up-to-date survey of around 150 P-complete problems.

- *Alternating graph accessibility.* Given is a directed graph with a set of vertices V and a set of edges E, a subset U of V, and designated vertices a and b in V. The vertices in U are called *universal* and those in $V \setminus U$ are called *existential*.

  The problem is to decide if $apath(a, b)$ holds, where, for any two vertices x and y, $apath(x, y)$ is true if either

  1. $x = y$, or
  2. x is existential and there exists a vertex z with $(x, z) \in E$ and $apath(z, y)$ is true, or
  3. x is universal and for all vertices z with $(x, z) \in E$, $apath(z, y)$ is true.

- *Generability.* Given is a finite set Q, (the graph of) a binary function $\boldsymbol{f}$ on Q, a subset V of Q and an element q in Q.

  The problem is to decide if q is in the smallest subset of Q that includes V and is closed under $\boldsymbol{f}$.

The generability problem remains in P even with more than one function. More generally, it is the problem of deciding if, given a finite algebra, a subset of its universe and an element in it, this element is in the subalgebra generated by the given subset [89]. Actually, generability is basically the same problem as non-emptiness of DTAs. In the following proof it is easily seen that all reductions can be carried out within logarithmic space, assuming reasonable representations of the problems, and we do not mention that explicitly.

Theorem 5.1 *Non-emptiness of DTTAs, DTAs and TAs is P-complete.*

Proof. First we show how alternating graph accessibility reduces to non-emptiness of DTTAs. Consider a directed graph $G = (V, E)$ a subset U of V of universal vertices, and two designated vertices a and b in V. We can assume without loss of generality that the out-degree of any vertex in G is either two or zero. Le A be the TTA $(V, \Sigma, R, \{a\})$, where $\Sigma = \{c, g_1, g_2, f\}$, c is a constant, g_1, g_2 unary function symbols, and f a binary function symbol. Let the rules of A be as follows:

1. $b \longrightarrow_R c$,
2. for each vertex x and edges $(x, y_1), (x, y_2) \in E$,
   (a) if x is universal then $x \longrightarrow_R f(y_1, y_2)$,

(b) if x is existential then $x \longrightarrow_R g_1(y_1)$ and $x \longrightarrow_R g_2(y_2)$.

Clearly A is a DTTA. It follows easily that for any vertex x,

$$apath(x,b) \quad \Leftrightarrow \quad (\exists \tau \in \mathcal{T}_\Sigma)\ x \xrightarrow{*}_R \tau, \tag{5.3}$$

and thus $apath(a,b)$ iff $T(A)$ is non-empty. The '$\Rightarrow$' direction follows by induction on the size of any alternating path to b and case analysis on x (universal or existential). The base case ($x = b$) is trivial. Let us consider one induction case, namely, when x is existential and different from b. Then, for some vertex z,

$$\begin{array}{rcl} apath(x,b) & \Rightarrow & (x,z) \in E,\ apath(z,b) \\ & \overset{\text{(IH)}}{\Rightarrow} & x \longrightarrow_R g(z),\ z \xrightarrow{*}_R \tau \\ & \Rightarrow & x \xrightarrow{*}_R g(\tau), \end{array}$$

where $\tau \in \mathcal{T}_\Sigma$ and g is either g_1 or g_2. The '$\Leftarrow$' direction also follows easily by induction on the length of reductions.

We prove now that the non-emptiness problem of TTAs (and thus TAs) is in P by giving a simple reduction from it to alternating graph accessibility. Let A be a TTA (Q, Σ, R, I). Assume without loss of generality that there is only one constant c in Σ and that I is a singleton set $\{q_0\}$. We construct a graph $G = (V, E)$ with designated vertices a and b and a subset U as the set of universal vertices as follows. Let $V = Q \cup U$ where U is the collection $\{\, u_t \mid q \to t \in R \,\} \cup \{u_c\}$ of new vertices. Let $a = q_0$ and $b = u_c$. Let

$$E \quad = \quad \{\, (q,u_t), (u_t,q_1), \ldots, (u_t,q_n) \mid q \to \underbrace{f(q_1,\ldots,q_n)}_{t} \in R \,\}.$$

Like above, statement (5.3) is proved for all $x \in Q$ by induction. It follows that $apath(a,b)$ iff $T(A)$ is non-empty.

Finally, we give a simple reduction from generability to the non-emptiness problem of DTAs to show that it is P-hard. Let Q be a finite set, $\boldsymbol{f}$ a binary function on Q, $V \subseteq Q$ and $q_{\mathrm{f}} \in Q$. Let A be the DTA $(Q, \Sigma, R, \{q_{\mathrm{f}}\})$, where Σ consists of a binary function symbol f and a constant c_q for each $q \in V$. Let R be the following set of rules:

$$R \quad = \quad \{\, c_q \to q \mid q \in V \,\} \cup \{\, f(q_1,q_2) \to q \mid \boldsymbol{f}(q_1,q_2) = q \,\}.$$

It follows easily that $T(A)$ is non-empty iff q_{f} is in the least subset of Q including V that is closed under $\boldsymbol{f}$. ⊠

Non-emptiness of DTAs is in fact the same problem as (the more general formulation of) generability given above. Consider a total DTA A with

signature Σ as the pair $(\mathfrak{A}, F)$ where $\mathfrak{A}$ is a Σ-algebra and F a subset of its universe. Non-emptiness of $T(A)$ is simply the question of whether there exists a term $\tau \in \mathcal{T}_\Sigma$ such that $\tau^{\mathfrak{A}} \in F$, or in other words, if the subalgebra of $\mathfrak{A}$ generated by the empty set intersects with F.

The non-emptiness problem is clearly a particular case of the inequivalence problem. It is also easy to see that there is logspace reduction from any two DTAs A and B to the DTA in (5.2). It follows thus that inequivalence of DTAs is also P-complete. From a statement in Seidl [132, Theorem 4.3] follows that inequivalence of DTTAs is P-complete as well. For TAs in general the situation is different, however. In order to reduce the inequivalence problem of two TAs into the non-emptiness problem by using (5.2) it is necessary to first transform the TAs in question into DTAs which in general implies an exponential increase in the number of states (this is true already in the case of NFAs [122, 101]). In fact, Seidl has proved that the inequivalence problem of TAs is EXPTIME-complete [132, Theorem 2.1]. The inequivalence problem of NFAs and regular expressions is PSPACE-complete [102]. For more recent developments regarding complexity of word problems see Mayer and Stockmeyer [100].

5.5 INTERSECTION NON-EMPTINESS

EXPTIME-hardness of the intersection non-emptiness problem of finite tree automata has been observed by other researchers and used in various contexts. It was first remarked by Frühwirth et al [48] and used in the context of type inference of logic programs. Goubault gives an incomplete proof in the case of DTAs in the context of a faulty proof of EXPTIME-completeness of SREU [66]. Seidl [133] uses EXPTIME-hardness of the intersection non-emptiness of DTTAs and outlines a proof, in the context of sort inference in typed functional programming. The proof presented here is a generalization of the proof of PSPACE-hardness of the intersection non-emptiness of DFAs by Kozen [90]. The idea is to encode the set of valid computation trees of a fixed polynomial space ATM and a given input string, as the forest given by the intersection of a collection of DTAs. The same idea is used in the above references.

To see that the intersection non-emptiness problem can be solved in exponential time consider a sequence $(A_i)_{i<n}$ of TAs and take their direct product, call it A. Clearly A can be constructed in exponential time and we know also that $\bigcap_{i<n} T(A_i) = T(A)$. So, inclusion in EXPTIME follows by Theorem 5.1. Without using Theorem 5.1, one can reduce the intersection non-emptiness problem to other problems known to be in EXPTIME (or EXPTIME-complete), like the inference problem for full implicational dependencies [16], relational query evaluation [146] or a certain restricted

logic program [134] (using the relationship ALOGSPCE=P [17]). One such reduction is given below.

Theorem 5.2 *Intersection non-emptiness of TAs and DTAs is EXPTIME-complete.*

A formal proof of Theorem 5.2 is given in the subsequent sections as lemmas 5.4 and 5.8. The following outline illustrates the main ideas of that proof.

Proof. (Outline) Inclusion in EXPTIME is explained above. EXPTIME-hardness is proved as follows. Let $M = ((Q, \Sigma_{\text{in}}, \Sigma, \delta, q_0, \text{ƀ}, \{q_{\text{f}}\}), U)$ be a fixed polynomial space ATM and x a fixed input string. Let n be the maximum number of tape cells used by M and let ID denote the IDs of M of legth n (possibly padded with extra blanks). We can assume that M has a unique final ID describing an empty tape and that each universal ID has either 2 or 0 successors. Define a ternary relation $\rhd \subseteq ID \times ID \times (ID \cup \{\epsilon\})$ such that $v \rhd (v_1, v_2)$ iff

1. v is existential, $v \vdash_M v_1$ and $v_2 = \epsilon$, or
2. v is universal, $v \vdash_M v_1$, $v \vdash_M v_2$ and $v_1 \neq v_2$.

For each k, $1 \leq k \leq n$, imagine that we have a small window that provides us only with a restricted view of M's tape immediately surrounding the k'th symbol. Let $\rhd_k \subseteq ID \times ID \times (ID \cup \{\epsilon\})$ be such that $v \rhd_k (v_1, v_2)$ iff $v \rhd (v_1, v_2)$ is possible according to that view. In addition assume that

$$\bigwedge_{k=1}^{n} v \rhd_k (v_1, v_2) \quad \Leftrightarrow \quad v \rhd (v_1, v_2). \tag{5.4}$$

Let Γ be a signature consisting of $Q \cup \Sigma$ as unary function symbols, a binary function symbol $\langle\rangle$ and a constant *nil*. For any ID $v = c_1 c_2 \cdots c_{n-1} c_n$ and term τ we write τv as a shorthand for the term $c_n(c_{n-1}(\cdots c_2(c_1(\tau))\cdots))$, and for any two terms τ_1 and τ_2 we write $\langle \tau_1, \tau_2 \rangle$ for the term $\langle\rangle(\tau_1, \tau_2)$. Define *ID-trees* as the least class of terms in $\mathcal{T}_\Gamma$ that satisfies:

1. *nil* is the *empty* ID-tree;
2. if τ_1 and τ_2 are ID-trees such that either both are empty or only τ_2 is empty and $v \in ID$ then $\tau = \langle \tau_1, \tau_2 \rangle v\text{ƀ}$ is an ID-tree.

We refer to τ_1 and τ_2 as the *left* and *right subtrees* of τ, v is called the *root* of τ. Define also root of *nil* to be the empty string (ϵ). A *move-tree* is any ID-tree τ such that for each internal subtree τ' of τ,

$$\text{Root}(\tau') \rhd (\text{Root}(\text{Left}(\tau')), \text{Root}(\text{Right}(\tau'))).$$

A move-tree is called *valid* if its root is the initial ID and its leaves are final IDs.

The kernel of the proof is a polynomial time construction of tree automata A_k and A_k^* for each k, $1 \leq k \leq n$, recognizing the following forests. The automaton A_k recognizes the set of all ID-trees τ such that

1. each leaf of τ is a final ID, and
2. for each internal subtree at level m in τ, where m is *even*,

$$\mathrm{Root}(\tau') \rhd_k (\mathrm{Root}(\mathrm{Left}(\tau')), \mathrm{Root}(\mathrm{Right}(\tau'))). \tag{5.5}$$

The automaton A_k^* recognizes the set of all ID-trees τ such that

1. the root of τ is the initial ID, and
2. (5.5) holds for each internal subtree τ' at level m in τ, where m is *odd*.

It follows that $\tau \in \bigcap_{k=1}^{n} T(A_k) \cap T(A_k^*)$ iff τ is a valid move-tree. ⊠

Any signature can easily be encoded with just one binary function symbol and a collection of constants. The following corollary is an easy consequence.

Corollary 5.1 *Intersection non-emptiness of DTAs when restricted to signatures with one binary function symbol and constants is EXPTIME-hard.*

EXPTIME-hardness

We give a polynomial time reduction of polynomial space ATMs to the intersection non-emptiness problem of DTAs. It follows that the problem is APSPACE-hard and thus EXPTIME-hard. For the rest of this section let

$$M = ((Q, \Sigma_{\mathrm{in}}, \Sigma, \delta, q_0, \text{ƀ}, F), U)$$

be a fixed ATM that is space-bounded by some polynomial S such that $S(m) \geq m$. We can assume, without loss of generality, that M has a single tape, this follows from a straightforward generalization of the corresponding property for TMs [76, Theorem 12.2]. Let $x \in \Sigma_{\mathrm{in}}^+$ be a fixed string and $n = S(|x|)$. Let *ID* stand for the set of all possible strings that represent IDs of M that may be padded with extra blanks at the end so that each string represents the first n tape symbols of M, i.e.,

$$ID \quad = \quad \bigcup_{0 \leq k < n} \Sigma^{(k)} Q \Sigma^{(n-k)}.$$

From here on we say ID for any element of *ID*. We can assume without loss of generality that M satisfies the following conditions:

- The initial state q_0 is existential and occurs only in the initial ID ($ID_0 = q_0 x \natural^{(n-|x|)}$).
- M has exactly one final state q_f and the final ID has the form $ID_f = q_f \natural^{(n)}$.
- Each universal ID has 0 or 2 successors.

Let all the symbols in $\Sigma \cup Q$ have arity 1, i.e., treat them like unary function symbols. Let also $\langle\rangle$ and *nil* be new function symbols with arities 2 and 0, respectively. Let $\Gamma = \Sigma \cup Q \cup \{\langle\rangle, nil\}$. We represent "computations trees" of M by certain terms in $\mathcal{T}_\Gamma$. For a string $v = c_1 c_2 \cdots c_m$ over $\Sigma \cup Q$ and τ a term we write τv for the term $c_m(c_{m-1}(\cdots c_1(\tau)\cdots))$, and for any two terms τ_1 and τ_2 we write $\langle \tau_1, \tau_2 \rangle$ for the term $\langle\rangle(\tau_1, \tau_2)$.

▶ *ID-trees* is the least class of terms in $\mathcal{T}_\Gamma$ that satisfies:

- *nil* is an ID-tree, called the *empty* ID-tree;
- if τ_1 and τ_2 are ID-trees such that either both are empty or only τ_2 is empty and $v \in ID$ then $\tau = \langle \tau_1, \tau_2 \rangle v \natural$ is an ID-tree.

We refer to τ_1 and τ_2 as the *left* and *right subtrees* (or collectively *immediate subtrees*) of τ, v is called the *root* of τ. We use the notations $\mathrm{Left}(\tau)$, $\mathrm{Right}(\tau)$ and $\mathrm{Root}(\tau)$. We let also $\mathrm{Root}(nil) = \epsilon$.

Let τ and τ' be ID-trees. We say that τ' is an *m-fold subtree* of τ if either $m = 0$ and $\tau' = \tau$ or τ' is an $(m-1)$-fold subtree of some immediate subtree of τ. By *subtree* we mean m-fold subtree for some $m \geq 0$. The *depth* of τ is the largest $m \geq 0$ such that there exists an m-fold subtree of τ, e.g., the depth of *nil* is 0.

The roots of all the non-empty subtrees of τ are called its *nodes*. A non-empty subtree of τ with empty immediate subtrees is called *external*. A non-empty subtree of τ that is not external is called *internal*. The root of any external subtree of τ is called a *leaf* of τ. Below we use the following definitions.

▶ An *ID-triple* is any element of $ID \times ID \times (ID \cup \{\epsilon\})$, where ϵ denotes the empty string. By a *move* of M we mean any ID-triple (v, v_1, v_2) where either

- v is existential, $v \vdash v_1$ and $v_2 = \epsilon$, or
- v is universal, $v \vdash v_1$, $v \vdash v_2$ and $v_1 \neq v_2$.

We write $v \rhd (v_1, v_2)$ iff (v, v_1, v_2) is a move.

- A *move-tree* is any ID-tree τ such that for each internal subtree τ' of τ,
$$\mathrm{Root}(\tau') \rhd (\mathrm{Root}(\mathrm{Left}(\tau')), \mathrm{Root}(\mathrm{Right}(\tau'))).$$
A move-tree is *valid* if its root is the initial ID and its leaves are final IDs.

The notion of a valid move-tree is a straightforward generalization of the notion of a valid computation of M on input x. We exploit the following obvious characterization of acceptance in terms of valid move-trees: *M accepts x iff there exists a valid move-tree.*

Main Construction The kernel of the hardness proof is a polynomial time construction of a collection of tree automata such that their intersection is precisely the set of all valid move-trees. We construct two kinds of automata, one for each k, $1 \leq k \leq n$.

1. The first kind recognizes all move-trees the leaves of which are final IDs and which satisfy the following additional property. Roughly, for all internal m-fold subtrees τ where m is *even*, the ID-triple (v, v_1, v_2), where v is the root of τ and v_1 and v_2 the roots of the left and right subtrees of τ, is a possible move by looking only at the tape symbols immediately surrounding the k'th symbol.
2. The second kind recognizes all move-trees the root of which is the initial ID and which satisfy the same additional property as above, except for *odd* m.

First, we formally define the sets of ID-trees correspeonding to items 1 and 2, and show that their intersection gives us precisely all the valid move-trees. Then we present formal constructions of DTAs that recognize these sets. We need some additional notations and definitions.

By a *position* we mean any integer k such that $1 \leq k \leq n$. Let k be a position and $v = a_1 \cdots a_{i-1} q a_i \cdots a_n \in ID$ where $q \in Q$. We write $v[k]$ and $\mathrm{View}(v, k)$ for the following substrings of v,

$$v[k] = \begin{cases} qa_k, & \text{if } k = i; \\ a_k, & \text{otherwise.} \end{cases}$$

$$\mathrm{View}(v,k) = \begin{cases} v[k]v[k+1], & \text{if } k = 1; \\ v[k-1]v[k], & \text{if } k = n; \\ v[k-1]v[k]v[k+1], & \text{otherwise.} \end{cases}$$

We let also $\mathrm{View}(\epsilon, k) = \epsilon$ and for any ID-triple (v, v_1, v_2),

$$\mathrm{View}((v, v_1, v_2), k) = (\mathrm{View}(v,k), \mathrm{View}(v_1,k), \mathrm{View}(v_2,k)).$$

Consider a fixed position k.

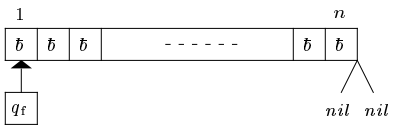


Figure 5.1: Base case of T_k.

- A *k-move* is an ID-triple $\vec{v} = (v, v_1, v_2)$ such that the following holds.
  1. If $v[k] \in Q\Sigma$ then there exists a move $\vec{w}$ such that $\text{View}(\vec{w}, k) = \text{View}(\vec{v}, k)$.
  2. If $v[k] = a \in \Sigma$ then $v_1[k] \in \{a\} \cup Qa$ and either $v_2 = \epsilon$ or $v_2[k] \in \{a\} \cup Qa$.

  We write $v \rhd_k (v_1, v_2)$ iff (v, v_1, v_2) is a k-move.

The following lemmas follow easily and we leave their proofs to the reader.

Lemma 5.1 *An ID-triple is a move iff it is a k-move for all positions k.*

Lemma 5.2 *For all positions k and all ID-triples $\vec{v}$ and $\vec{w}$. If $\vec{v}$ is a k-move and* $\text{View}(\vec{v}, k) = \text{View}(\vec{w}, k)$ *then $\vec{w}$ is a k-move.*

For all positions k, let T_k denote the following set of terms. Below we show that T_k is recognizable, and that the time complexity to construct a tree automaton that recognizes T_k, is polynomial in n.

- T_k is the set T of all *ID*-trees such that
  1. $\langle nil, nil\rangle ID_f ƀ \in T$,
  2. $\langle \tau_1, \tau_2\rangle v ƀ \in T$ if τ_1 is non-empty and,
     (a) $v \rhd_k (\text{Root}(\tau_1), \text{Root}(\tau_2))$,
     (b) $\text{Left}(\tau_1) \in T$ and $\text{Right}(\tau_1) \in T \cup \{nil\}$, and
     (c) either τ_2 is empty, or $\text{Left}(\tau_2) \in T$ and $\text{Right}(\tau_2) \in T \cup \{nil\}$.

So any *ID*-tree in T_k has external subtrees of the form shown in Figure 5.1. A possible induction case is illustrated in Figure 5.2. For each position k, we let T_k^* denote the following sets of terms. Also in this case we show that each T_k^* is recognizable by a tree automaton that can be constructed in polynomial time.

- T_k^* is the set of all $\langle \tau_1, \tau_2\rangle ID_0 ƀ$ where $\tau_1, \tau_2 \in T$ and either both are empty or only τ_2 is empty, where T is the set of all *ID*-trees where q_0 does not occur such that

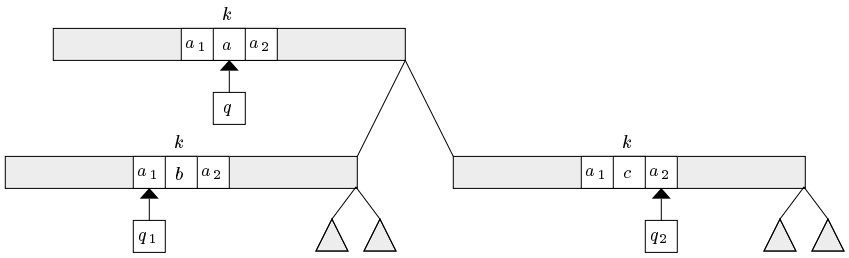


Figure 5.2: One possible induction case of T_k; q is universal and $\delta(q, a) = \{(q_1, b, \texttt{left}), (q_2, c, \texttt{right})\}$.

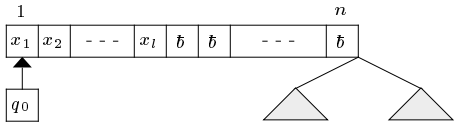


Figure 5.3: All *ID*-trees in T_k^* have this form.

1. $nil \in T$,
2. $\langle \tau_1, \tau_2 \rangle v\flat \in T$ if τ_1 is non-empty and (a–c) hold,
   (a) $v \rhd_k (\mathrm{Root}(\tau_1), \mathrm{Root}(\tau_2))$,
   (b) $\mathrm{Left}(\tau_1), \mathrm{Right}(\tau_1) \in T$, and
   (c) either τ_2 is empty or $\mathrm{Left}(\tau_2), \mathrm{Right}(\tau_2) \in T$.

All *ID*-trees in T_k^* are illustrated in Figure 5.3.

Let $\tau \in T_k \cap T_k^*$ and let τ' be any internal m-fold subtree of τ for some $m \geq 0$. If m is even (odd) then it follows by definition of T_k (T_k^*), that

$$\mathrm{Root}(\tau') \rhd_k (\mathrm{Root}(\mathrm{Left}(\tau')), \mathrm{Root}(\mathrm{Right}(\tau'))).$$

We have thus the following property.

Lemma 5.3 *For all positions k, if $\tau \in T_k \cap T_k^*$ then for all internal subtrees τ' of τ,*

$$\mathrm{Root}(\tau') \rhd_k (\mathrm{Root}(\mathrm{Left}(\tau')), \mathrm{Root}(\mathrm{Right}(\tau'))).$$

We can now state our main lemma.

Lemma 5.4 *Intersection non-emptiness of DTAs is EXPTIME-hard.*

Proof. Construct tree automata A_k and A_k^* for $1 \leq k \leq n$ such that $T(A_k) = T_k$ and $T(A_k^*) = T_k^*$ (see Lemma 5.5 and Lemma 5.6). Each one

is constructed in time that is polynomial in $n = S(|x|)$, and thus the total time complexity of the construction of all the automata is polynomial in $|x|$. It is sufficient to show that

$$\{\, \tau \mid \tau \text{ is a valid move-tree} \,\} = \bigcap_{k=1}^{n} (T_k \cap T_k^*)$$

The direction '$\subseteq$' (i.e., that each valid move tree is in T_k and T_k^*) is easy to check. (Note that the property that all computation paths of M have even length is needed here.) We prove the direction '$\supseteq$'. Let $\tau \in \bigcap_{k=1}^{n} (T_k \cap T_k^*)$. It follows immediately from the definition of any T_k that the leaves of τ are final IDs. It follows also immediately from the definition of any T_k^* that the root of τ is the initial ID. It remains to prove that τ is a move-tree, i.e., that for any internal subtree τ' of τ,

$$\mathrm{Root}(\tau') \rhd (\mathrm{Root}(\mathrm{Left}(\tau')), \mathrm{Root}(\mathrm{Right}(\tau'))),$$

but this follows by first applying Lemma 5.3 and then Lemma 5.1. ⊠

Recognizability of T_k Consider a fixed position k distinct from 1 and n. The handling of positions 1 and n is similar. We construct a tree automaton A_k that recognizes T_k. It is clear that one can easily extract an algorithm from this construction that has polynomial time complexity in n. Let

$$\begin{aligned} \Delta &= \Sigma\Sigma\Sigma \cup Q\Sigma\Sigma\Sigma \cup \Sigma Q\Sigma\Sigma \cup \Sigma\Sigma Q\Sigma, \\ I &= \Delta \times (\Delta \cup \{\epsilon\}). \end{aligned}$$

As the main part in the construction of A_k we use a family $\{M_i\}_{i \in I \cup \{0\}}$ of DFAs, where each M_i is a DFA that accepts ID and for each $v \in ID$ simply scans v and accepts it in the final state $p_{(\alpha,i)}$ iff $\mathrm{View}(v,k) = \alpha$. Formally, for all $i \in I \cup \{0\}$,

$$M_i = (P_i, \Sigma \cup Q, \delta_i, p_{(0,i)}, \{\, p_{(\alpha,i)} \mid \alpha \in \Delta \,\}), \quad L(M_i) = ID,$$

such that for all $\alpha \in \Delta$ and $v \in ID$,

$$\delta_i(p_{(0,i)}, v) = p_{(\alpha,i)} \quad \Leftrightarrow \quad \mathrm{View}(v,k) = \alpha.$$

Furthermore, all the P_i's are assumed to be pairwise disjoint. In particular we can take all the members to be copies of say M_0. It is easy to construct M_0 in time that is polynomial in n. Let also M_{f} be a DFA (with new states) such that

$$M_{\mathrm{f}} = (P_{\mathrm{f}}, \Sigma \cup Q, \delta_{\mathrm{f}}, p_{(0,\mathrm{f})}, \{p_{\mathrm{f}}\}), \quad L(M_{\mathrm{f}}) = \{ID_{\mathrm{f}}\}.$$

Let now R_i for $i \in I \cup \{0, \mathrm{f}\}$ denote following sets of rules:

$$R_i \quad = \quad \{\, c(p) \to p' \mid \delta_i(c,p) = p',\ c \in \Sigma \cup Q,\ p, p' \in P_i \,\}.$$

For any string $v = c_1 c_2 \ldots c_{m-1} c_m$ over $\Sigma \cup Q$ and state p we write pv for the term $c_m(c_{m-1}(\cdots c_2(c_1(p))\cdots))$. It is clear that for any string v over $\Sigma \cup Q$, and any two states p and p' in P_i,

$$\delta_i(p, v) = p' \quad \Leftrightarrow \quad pv \xrightarrow{*}_{R_i} p'.$$

Let $\{t_\epsilon, t_\mathrm{f}\} \cup \{\, t_\alpha \mid \alpha \in \Delta \,\}$ be a set of new state symbols.

- A_k is the following tree automaton:

$$\begin{aligned}
Q^{A_k} &= \{t_\epsilon, t_\mathrm{f}\} \cup \{\, t_\alpha \mid \alpha \in \Delta \,\} \cup P_0 \cup P_\mathrm{f} \cup \bigcup_{i \in I} P_i, \\
\Sigma^{A_k} &= \Gamma, \\
R^{A_k} &= \bigcup_{i \in I} R_i \cup R_0 \cup R_\mathrm{f} \cup \\
&\quad \{\, nil \to t_\epsilon \,\} \cup \\
&\quad \{\, \langle t_\epsilon, t_\epsilon \rangle \to p_{(0,\mathrm{f})} \,\} \cup \\
&\quad \{\, \langle t_\mathrm{f}, t_\epsilon \rangle \to p_{(0,0)},\ \langle t_\mathrm{f}, t_\mathrm{f} \rangle \to p_{(0,0)} \} \cup \\
&\quad \{\, \langle t_\beta, t_\gamma \rangle \to p_{(0,\beta,\gamma)} \mid (\beta, \gamma) \in I \,\} \cup \\
&\quad \{\, p_{(\alpha,0)} \natural \to t_\alpha \mid \alpha \in \Delta \,\} \cup \\
&\quad \{\, p_{\mathrm{View}((v,v_1,v_2),k)} \natural \to t_\mathrm{f} \mid v \rhd_k (v_1, v_2) \,\} \cup \\
&\quad \{\, p_\mathrm{f} \natural \to t_\mathrm{f} \,\}, \\
F^{A_k} &= \{\, t_\mathrm{f} \,\}.
\end{aligned}$$

Note that $p_{\mathrm{View}((v,v_1,v_2),k)}$ is the final state $p_{(\mathrm{View}(v,k),i)}$ in M_i, where i is the index $(\mathrm{View}(v_1,k), \mathrm{View}(v_2,k))$. It is easy to check that A_k is indeed a deterministic tree automaton. The structure of A_k is illustrated in Figure 5.4.

Lemma 5.5 $T(A_k) = T_k$.

Proof.

[**Proof of** $T(A_k) \subseteq T_k$] Let $\tau \in T(A_k)$, i.e., $\tau \in \mathcal{T}_\Gamma$ and $\tau \xrightarrow{*}_{R^{A_k}} t_\mathrm{f}$. We prove that $\tau \in T_k$. The proof is by induction on the length of the reduction $\tau \xrightarrow{*} t_\mathrm{f}$. There are two cases, depending on the last step of the reduction.

1. $\tau \xrightarrow{*} p_\mathrm{f} \natural \longrightarrow t_\mathrm{f}$, or
2. $\tau \xrightarrow{*} p_{\mathrm{View}(\vec{v},k)} \natural \longrightarrow t_\mathrm{f}$ for some k-move $\vec{v}$. Let $\mathrm{View}(\vec{v}, k) = (\alpha, \beta, \gamma)$.

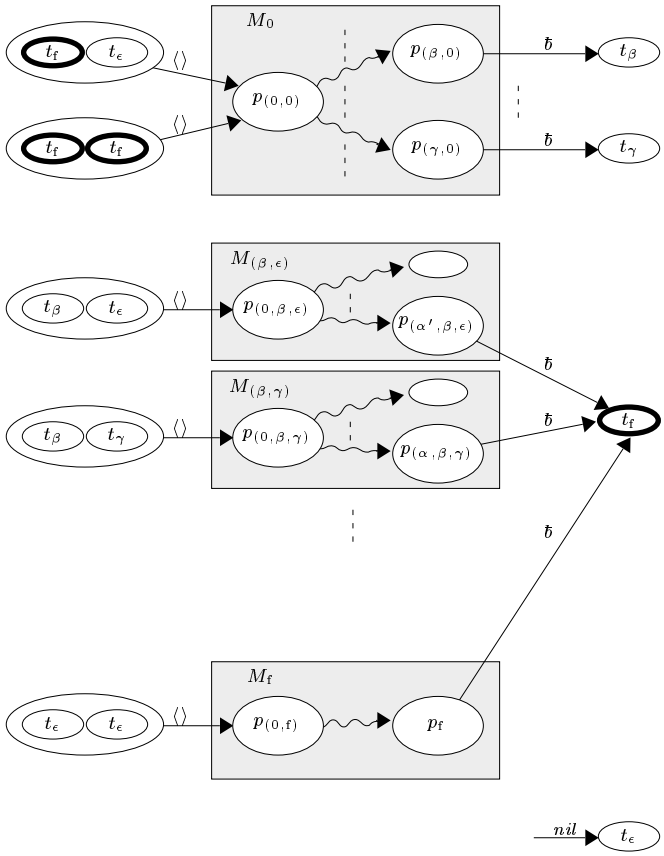


Figure 5.4: Tree automaton A_k. A transition from $p_{(\alpha,\beta,\gamma)}$ to t_f exists only if $(\alpha,\beta,\gamma) = \mathrm{View}(\vec{v},k)$ for some k-move $\vec{v}$.

Let us consider the first case first. From the definition of the rules of A^k and the disjointness of the underlying DFAs it follows that the reduction has to be of the following form: (simply trace the arrows backwards in Figure 5.4)

$$\begin{aligned} p_\mathrm{f}\flat &\longrightarrow t_\mathrm{f} \\ p_{(0,\mathrm{f})}ID_\mathrm{f} &\xrightarrow{*}_{R_\mathrm{f}} p_\mathrm{f} \\ \langle nil, nil\rangle &\xrightarrow{*} p_{(0,\mathrm{f})}, \end{aligned}$$

which shows that $\tau = \langle nil, nil\rangle ID_\mathrm{f}\flat$, and thus $\tau \in T_k$. We now consider the second case. Then

$$\begin{aligned} p_{(\alpha,\beta,\gamma)}\flat &\longrightarrow t_\mathrm{f} \\ p_{(0,\beta,\gamma)}w &\xrightarrow{*}_{R_{(\beta,\gamma)}} p_{(\alpha,\beta,\gamma)} \quad (\text{some } w \in ID \text{ such that } \mathrm{View}(w,k) = \alpha) \end{aligned}$$

$$\langle t_\beta, t_\gamma \rangle \quad \longrightarrow \quad p_{(0,\beta,\gamma)}.$$

So $\tau = \langle \tau_1, \tau_2 \rangle w \natural$ where $\tau_1 \xrightarrow{*} t_\beta$ and $\tau_2 \xrightarrow{*} t_\gamma$. Since $\beta \neq \epsilon$ it follows that the reduction $\tau_1 \xrightarrow{*} t_\beta$ must have the following form:

$$\begin{array}{lll} p_{(\beta,0)}\natural & \longrightarrow & t_\beta \\ p_{(0,0)} w_1 & \xrightarrow{*}_{R_0} & p_{(\beta,0)} \quad \text{(some } w_1 \in ID \text{ such that View}(w_1, k) = \beta\text{)}, \end{array}$$

and either $\langle t_{\rm f}, t_\epsilon \rangle \longrightarrow p_{(0,0)}$ or $\langle t_{\rm f}, t_{\rm f} \rangle \longrightarrow p_{(0,0)}$. Assume (without loss of generality) that the former reduction step took place and that $\gamma = \epsilon$ (and thus $\tau_2 = nil$).

Under these conditions $\mathrm{Root}(\tau_1) = w_1$, $\mathrm{Left}(\tau_1) \xrightarrow{*} t_{\rm f}$ and $\mathrm{Right}(\tau_1) = nil$. It follows by the induction hypothesis that $\mathrm{Left}(\tau_1) \in T_k$. Let $\vec{w} = (w, w_1, \epsilon)$, since $\mathrm{View}(\vec{w}, k) = \mathrm{View}(\vec{v}, k)$ and $\vec{v}$ is a k-move, it follows by Lemma 5.2 that $\vec{w}$ is a k-move. Now $\tau \in T_k$ by the definition of T_k.

[**Proof of** $T_k \subseteq T(A_k)$] Let $\tau \in T_k$. Clearly $\tau \in \mathcal{T}_\Gamma$. We must show that $\tau \xrightarrow{*} t_{\rm f}$. The proof is by induction on the size of τ. The base case is $\tau = \langle nil, nil \rangle ID_{\rm f} \natural$ and it follows by above that $\tau \xrightarrow{*} t_{\rm f}$. The induction case is $\tau = \langle \tau_1, \tau_2 \rangle v \natural$, where $\tau_1 = \langle \tau_{11}, \tau_{12} \rangle v_1 \natural$,

1. $v \rhd_k (v_1, \mathrm{Root}(\tau_2))$,
2. $\tau_{11} \in T_k$ and $\tau_{12} \in T_k \cup \{nil\}$, and
3. either τ_2 is empty or $\mathrm{Left}(\tau_2) \in T_k$ and $\mathrm{Right}(\tau_2) \in T_k \cup \{nil\}$.

We can assume, without loss of generality, that τ_2 and τ_{12} are empty. Let $(\alpha, \beta, \epsilon) = \mathrm{View}((v, v_1, \epsilon), k)$. By using the induction hypothesis and the rules of A_k we obtain the following reduction:

$$\begin{array}{rll} \tau & \xrightarrow{*}_{(\mathrm{IH})} & \langle \langle t_{\rm f}, t_\epsilon \rangle v_1 \natural, t_\epsilon \rangle v \natural \\ & \longrightarrow & \langle p_{(0,0)} v_1 \natural, t_\epsilon \rangle v \natural \\ & \xrightarrow{*}_{R_0} & \langle p_{(\beta,0)} \natural, t_\epsilon \rangle v \natural \\ & \longrightarrow & \langle t_\beta, t_\epsilon \rangle v \natural \\ & \longrightarrow & p_{(0,\beta,\epsilon)} v \natural \\ & \xrightarrow{*}_{R_{(\beta,\epsilon)}} & p_{(\alpha,\beta,\epsilon)} \natural. \end{array}$$

But $v \rhd_k (v_1, \epsilon)$, and thus $p_{(\alpha,\beta,\epsilon)} \natural \longrightarrow t_{\rm f}$. ⊠

Recognizability of T_k^* As above, we consider a fixed position k distinct from 1 and n, and construct a tree automaton A_k^* that recognizes T_k^*. It is clear that the construction can be carried out in time that is polynomial in n. We are not as detailed below as we are above, due to the similarity of the construction.

Let Δ and I be as in Section 5.5 except that the initial state q_0 of M is omitted from Q. Let also M_i for $i \in I \cup \{0\}$ have the same definition (except for that same restriction). Let M_f be the following DFA: (with new states)

$$M_\mathrm{f} = (P_\mathrm{f}, \Sigma, \delta_\mathrm{f}, p_{(0,\mathrm{f})}, \{p_\mathrm{f}\}), \quad L(M_\mathrm{f}) = \{x\bar{b}^{(n-|x|)}\}.$$

Let now R_i for $i \in I \cup \{0, \mathrm{f}\}$ denote the same sets of rules as defined above. Let $\{t, t_\epsilon, t_\mathrm{f}\} \cup \{\, t_\alpha \mid \alpha \in \Delta \,\}$ be a set of new state symbols.

- A_k^* is the following tree automaton:

$$\begin{aligned}
Q^{A_k^*} &= \{t, t_\epsilon, t_\mathrm{f}\} \cup \{\, t_\alpha \mid \alpha \in \Delta \,\} \cup P_0 \cup P_\mathrm{f} \cup \bigcup_{i \in I} P_i, \\
\Sigma^{A_k^*} &= \Gamma, \\
R^{A_k^*} &= \bigcup_{i \in I} R_i \cup R_0 \cup R_\mathrm{f} \cup \{q_0(p_{(0,0)}) \to p_{(0,\mathrm{f})}\} \cup \\
&\quad \{\, \mathit{nil} \to t_\epsilon \,\} \cup \\
&\quad \{\, \langle t_\epsilon, t_\epsilon\rangle \to p_{(0,0)},\ \langle t, t_\epsilon\rangle \to p_{(0,0)},\ \langle t, t\rangle \to p_{(0,0)} \,\} \cup \\
&\quad \{\, \langle t_\beta, t_\gamma\rangle \to p_{(0,\beta,\gamma)} \mid (\beta, \gamma) \in I \,\} \cup \\
&\quad \{\, p_{(\alpha,0)}\bar{b} \to t_\alpha \mid \alpha \in \Delta \,\} \cup \\
&\quad \{\, p_{\mathrm{View}((v,v_1,v_2),k)}\bar{b} \to t \mid v \rhd_k (v_1, v_2) \,\} \cup \\
&\quad \{\, p_\mathrm{f}\bar{b} \to t_\mathrm{f} \,\}, \\
F^{A_k^*} &= \{\, t_\mathrm{f} \,\}.
\end{aligned}$$

Note that A_k^* is indeed a deterministic tree automaton (in particular note that the q_0-transition from $p_{(0,0)}$ to $p_{(0,\mathrm{f})}$ does not violate the determinism). The structure of A_k^* is illustrated in Figure 5.5. The proof of Lemma 5.6 is analogous to the proof of Lemma 5.5.

Lemma 5.6 $T(A_k^*) = T_k^*$.

Inclusion in EXPTIME

We reduce the intersection non-emptiness problem of TAs to the *inference problem for full implicational dependencies* or FIDs. An FID is just a universal relational Horn sentence, we write it here as a "backward" implication

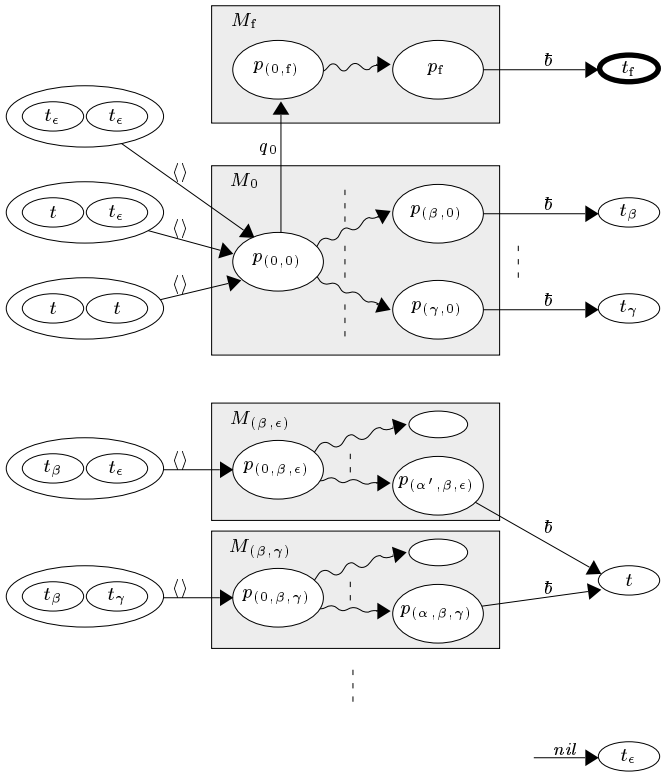


Figure 5.5: Tree automaton A_k^*. A transition from $p_{(\alpha,\beta,\gamma)}$ to t exists only if $(\alpha,\beta,\gamma) = \mathrm{View}(\vec{v},k)$ for some k-move $\vec{v}$.

$\varphi \leftarrow \psi$ where φ is an atom and ψ a conjunction of atoms. The only function symbols in an FID are constants. The inference problem is simply the question of whether a given conjunction of FIDs implies another given FID. This problem can be solved in exponential time (actually it is EXPTIME-complete [16, 146]).

Let A_i for $1 \leq i \leq n$ for some $n \geq 1$ be TAs with a common input alphabet Σ,

$$A_i = (Q_i, \Sigma, R_i, F_i), \quad (1 \leq i \leq n).$$

Let $A = (Q, \Sigma, R, F)$ be the direct product of all the A_i's. So the states of A are elements of $\prod_{i=1}^n Q_i$ and the rules of A are defined as follows, we write $\bar{q}$ for $(q_1, q_2, \ldots, q_n) \in Q$:

$$R \quad = \quad \{\, \sigma(\bar{q}_1, \ldots, \bar{q}_k) \to \bar{q} \mid \sigma(q_{i1}, q_{i2}, \ldots, q_{ik}) \longrightarrow_{R_i} q_i \quad (1 \leq i \leq n) \,\}.$$

We know that $T(A)$ is non-empty iff $\bigcap_{i=1}^n T(A_i)$ is non-empty. We construct a set of FIDs P with a distinguished atom `Nonempty` such that $P \vdash$ `Nonempty`

iff $T(A)$ is non-empty. Furthermore, it is obvious that this construction takes polynomial time in the total size of the A_i's (*not* in the size of A, the size of A is in general eponential in the total size of the A_i's).

First, for $1 \leq i \leq n$ and each k-ary function symbol $\sigma \in \Sigma$, let $\texttt{Rule}_i^\sigma$ be a new relation symbol of arity $k+1$. Let also $\texttt{Final}_i$ for $1 \leq i \leq n$ and $\texttt{Reduce}$ be relation symbols of arity 1 and n, respectively. To simplify matters, we can assume without loss of generality that all function symbols in Σ have arity at most 2. The following atoms (or atomic FIDs) are in P: for each A_i and final state q in it there is an atom

$$\texttt{Final}_i(q)$$

in P; for each A_i and rule $\sigma(q_1, \ldots, q_k) \to q$ in R_i (where $k \geq 0$), there is an atom

$$\texttt{Rule}_i^\sigma(q_1, \ldots, q_k, q)$$

in P. In addition, P includes the following FIDs: for each constant $\sigma \in \Sigma$ the FID

$$\texttt{Reduce}(\bar{x}) \leftarrow \bigwedge_{i=1}^{n} \texttt{Rule}_i^\sigma(x_i),$$

for each unary function symbol $\sigma \in \Sigma$ the FID

$$\texttt{Reduce}(\bar{x}) \leftarrow \bigwedge_{i=1}^{n} \texttt{Rule}_i^\sigma(y_i, x_i) \wedge \texttt{Reduce}(\bar{y})$$

and for each binary function symbol $\sigma \in \Sigma$ the FID

$$\texttt{Reduce}(\bar{x}) \leftarrow \bigwedge_{i=1}^{n} \texttt{Rule}_i^\sigma(y_i, z_i, x_i) \wedge \texttt{Reduce}(\bar{y}) \wedge \texttt{Reduce}(\bar{z}).$$

Finally, P includes the FID

$$\texttt{Nonempty} \leftarrow \texttt{Reduce}(\bar{x}) \wedge \bigwedge_{i=1}^{n} \texttt{Final}_i(x_i).$$

We have the following relationship between derivations from P and reduction in R.

Lemma 5.7 *For all $\bar{q} \in Q$, $P \vdash \texttt{Reduce}(\bar{q})$ iff there exists a term $\tau \in \mathcal{T}_\Sigma$ such that $\tau \xrightarrow{*}_R \bar{q}$.*

Proof. Let $\bar{q} \in Q$ be fixed and consider the direction '$\Rightarrow$'. Assume that $P \vdash \texttt{Reduce}(\bar{q})$. We prove by induction on the length of the proof of $P \vdash \texttt{Reduce}(\bar{q})$ that there exists a term $\tau \in \mathcal{T}_\Sigma$ such that $\tau \xrightarrow{*}_R \bar{q}$.

The base case is when there is a constant $c \in \Sigma$ such that $P \vdash \texttt{Rule}_i^c(q_i)$ for $1 \leq i \leq n$. Then $c \longrightarrow_{R_i} q_i$ for $1 \leq i \leq n$ and thus $\tau = c \longrightarrow_R \bar{q}$. The induction case is when there is a nonconstant function symbol $f \in \Sigma$ (we can assume that f is binary) and states $\bar{p}, \bar{r} \in Q$ such that

$$P \vdash \texttt{Rule}_i^f(p_i, r_i, q_i) \text{ (for } 1 \leq i \leq n), \quad P \vdash \texttt{Reduce}(\bar{p}), \quad P \vdash \texttt{Reduce}(\bar{r}).$$

By the induction hypothesis follows that there exist terms τ_1 and τ_2 in $\mathcal{T}_\Sigma$ such that $\tau_1 \longrightarrow_R \bar{p}$ and $\tau_2 \longrightarrow_R \bar{r}$. From $P \vdash \texttt{Rule}_i^f(p_i, r_i, q_i)$ (for $1 \leq i \leq n$) follows that $f(p_i, r_i) \longrightarrow_{R_i} q_i$ (for $1 \leq i \leq n$) and thus $f(\bar{p}, \bar{r}) \longrightarrow_R \bar{q}$. Consequently $\tau = f(\tau_1, \tau_2) \stackrel{*}{\longrightarrow}_R \bar{q}$.

The direction '$\Leftarrow$' is equally straightforward to prove by induction on the length of the reduction $\tau \stackrel{*}{\longrightarrow}_R \bar{q}$. ⊠

Since $P \vdash \texttt{Nonempty}$ iff there exists a final state $\bar{q}$ in A such that $P \vdash \texttt{Reduce}(\bar{q})$, it follows by Lemma 5.7 that $P \vdash \texttt{Nonempty}$ iff $T(A)$ is nonempty. The time to construct P is clearly polynomial in the total size of the A_i's. By Chandra et al. [16] it follows thus that:

Lemma 5.8 *The intersection non-emptiness problem of DTAs is in EXPTIME.*

We obtain an alternative proof of Lemma 5.8 by looking at P as a *logic program* and asking the question if the *goal* `Nonempty` follows from it. It is clear that in any proof tree of `Nonempty` from P the nodes (or intermediate goals) have a size that is linear in n, simply because there are no nonconstant function symbols in P. The computational complexity of the problem of deciding if $P \vdash \texttt{Nonempty}$ is therefore in EXPTIME by a correspondence between logic programs and ATMs by Shapiro [134, Theorem 4.4] and the relationship EXPTIME = APSPACE.

We can also note that NFAs correspond to *monadic* TAs, i.e., TAs over a signature where,besides constants, there are only unary function symbols. If we assume the above A_i's to be modadic then the non-emptiness problem of $T(A)$ corresponds to the non-emptiness problem of the intersection of the corresponding NFAs. It is easy to see by looking at P that one can construct an ATM without universal nodes (i.e., a TM) that uses only linear space in n and "accepts `Nonempty`" iff $P \vdash \texttt{Nonempty}$. Thus the intersection non-emptiness problem of NFAs is in PSPACE. This fact follows already from the proof of the PSPACE-completeness of the intersection non-emptiness problem of DFAs by Kozen [90], where the part of the proof regarding inclusion in PSPACE holds also for NFAs.

5.6 SUCCINCTNESS

The use of intersection can shorten a regular expression by an exponential amount. This fact explains why the inequivalence problem for regular expressions becomes EXPSPACE-complete when intersection is added [49, 79], whereas it is PSPACE-complete [102] in the usual case. (A similar effect is obtained with interleaving [100].) Above, we are just witnessing a similar effect on TAs. Namely, if we represent a TA A by a sequence of TAs with the same signature with the understanding that their product is A (modulo renaming of states), then the size of this representation can in some cases be exponentially more succinct than the size of A. With this notion of *succinctness* it follows that the intersection non-emptiness problem is simply the *succinct non-emptiness* problem. Note also that it is generally believed that EXPTIME is nothing else but P on exponentially more succinct input [112].

Analogously, succinct non-emptiness of finite automata is PSPACE-complete by Kozen's result [90]. If we consider a finite automaton as a graph, non-emptiness is just graph accessibility. In this case there is another notion of succinctness[4] which implies that the succinct graph accessibility problem is PSPACE-complete [113] (even for undirected graphs [94]).

5.7 CONCLUDING REMARKS

In this chapter we considered computational complexity of some basic decision problems of finite tree automata. In particular, we proved EXPTIME-completeness of the intersection non-emptiness problem (Theorem 5.2) and we showed P-completeness of the non-emptiness problem (Theorem 5.1). It follows that for a fixed number of finite tree automata, the problem of non-emptiness of their intersection is also P-complete. We discussed a notion of succinctness with respect to which the intersection non-emptiness problem is in fact a succinct version of the non-emptiness problem.

Our main motivation for studying these problems and their computational complexity is their close connection with the decidability and computational complexity of SREU with one variable. The computational complexities of the problems studied in this chapter and of closely related problems is summarized in Table 5.1. In general there seems to be a rule of thumb that says that *if a decision problem for (deterministic) finite automata is complete for a certain space complexity then the same decision problem with (deterministic) finite* tree *automata is complete for the corresponding deterministic time*

[4] The standard notion of succinctness in the case of graphs arises from practical considerations in VLSI [59]. A *succinct representation* of a graph G with n vertices is a Boolean circuit, that given (binary representations of) two integers $\leq n$ as input (representing two vertices in G), computes the corresponding entry of the adjacency matrix of G.

	Non-emptiness	Inequivalence	Intersection non-emptiness
DFA	NL	NL	PSPACE
NFA	NL	PSPACE	PSPACE
DTA	P	P	EXPTIME
DTTA	P	P	EXPTIME
TA	P	EXPTIME	EXPTIME

Table 5.1: Computational complexities of some basic decision problems of finite automata and finite tree automata. All problems are complete for the respective classes.

complexity, only one exponential higher. Besides Table 5.1, further justification for this rule follows by comparing computational complexities of some other decision problems of finite tree automata (studied by Seidl [132]) with the corresponding decision problems of finite automata studied by (Stearns and Hunt III [140, 141]). This relationship between computational complexities of decision problems of finite tree automata and finite automata is reflected by the fact that proofs of the former are usually extensions of proofs of the latter, by going from using nondeterministic Turing machines to using alternating Turing machines.

Remarks about Table 5.1 The non-emptiness problem of finite automata is in fact the graph accessibility problem and is thus complete for nondeterministic logarithmic space or NL-complete [128]. Using (5.2), inequivalence of DFAs reduces to non-emptiness [105] and since non-emptiness is a particular case of inequivalence, it follows that inequivalence of DFAs is NL-complete as well. For finite automata in general, inequivalence is PSPACE-complete by Meyer and Stockmeyer [102]. PSPACE-completeness of non-emptiness of intersection of finite automata was proved by Kozen [90].

Non-emptiness of finite tree automata is closely related to the two well-known P-complete problems: alternating graph accessibility [80] and generability [82, 89]. It follows by (5.2) that inequivalence of DTAs is also P-complete. EXPTIME-hardness of the intersection non-emptiness problem of finite tree automata has been observed by other researchers [48, 66, 133]. In particular, Seidl outlines a proof in the case of DTTAs [133]. He has also proved that inequivalence of TAs is EXPTIME-complete [132, Theorem 2.1] and it follows also from a statement by Seidl that when restricted to DTTAs, inequivalence is P-complete [132, Theorem 4.3].

SREU with One Variable

6.1 INTRODUCTION

We show that SREU with one variable is decidable. Moreover, we show that this problem is EXPTIME-complete. We prove also that rigid E-unification with one variable is P-complete and that SREU with one variable and a constant bound on the number of rigid equations is P-complete. One conclusion we draw from this is that the intractability of SREU with one variable is strongly related to the *number* of rigid equations and not their *size*. Note that with *two* variables, SREU is undecidable already with *three* rigid equations. Finally, we consider a case of SREU where one allows several variables, but each rigid equation either contains one variable, or has a ground left-hand side and an equality between two variables as a right-hand side. We show that SREU is decidable also in this restricted case. In Chapter 7 we use some of these results to obtain new decidability results in intuitionistic logic.

6.2 PRELIMINARIES

Recall that a *canonical model* of a set of ground equations E, denoted by $\mathcal{T}_{/E}$, is the quotient of $\mathcal{T}_\Sigma$ over $=_E$ where $=_E$ is simply the congruence relation induced by E over the set of ground terms over Σ. It is a simple fact that for all $s, t \in \mathcal{T}$,

$$\mathcal{T}_{/E} \models s \approx t \quad \Leftrightarrow \quad E \models s \approx t.$$

Structures that are isomorphic with the canonical model of a finite set of ground equations are also called *finitely presented algebras*. Various problems that are related to finitely presented algebras, and their computational complexity, have been studied in Kozen [88, 89]. Below, we make use of some of those results.

6.3 DECIDABILITY

In this section we establish formally the decidability of SREU with one variable. The proof has two parts.

1. First we prove that rigid E-unification with one variable can be reduced to the problem of testing membership in a finite union of congruence classes.
2. By using the property that any finite union of congruence classes is recognizable, we then reduce SREU with one variable to the intersection non-emptiness problem of finite tree automata.

The decidability of SREU with one variable follows then from the fact that recognizable sets are closed under Boolean operations and that the non-emptiness problem of finite tree automata is decidable. In Section 6.4 we address the computational complexity of this reduction.

Initial Reduction

We start by proving two lemmas. Roughly, these lemmas allow us to reduce an arbitrary rigid equation $S(x)$ with one variable to a finite collection of rigid equations $\{ S_i(x) \mid i < n \}$ such that, for all substitutions θ, θ solves S iff θ solves some S_i. Furthermore, each of the S_i's has the form $E \models_? x = t_i$ where E is ground and t_i is some ground term. The set E is common to all the S_i's.

Let E be a set of ground equations and t a ground term. Denote by $[t]_E$ the interpretation of t in $\mathcal{T}_{/E}$, in other words $[t]_E$ is the congruence class induced by $=_E$ on $\mathcal{T}$ that includes t. For a set T of ground terms we write $[T]_E$ for $\{ [t]_E \mid t \in T \}$. We write $\mathit{Terms}(E)$ for the set of all terms that occur in E, in particular $\mathit{Terms}(E)$ is closed under the subterm relation. We use the following lemma. Lemma 6.1 follows also from a more general statement in de Kogel [27, Theorem 5.11].

Lemma 6.1 *Let t be a ground term, c a constant, E a finite set of ground equations and e a ground equation. Let $T = \mathit{Terms}(E \cup \{e\})$. If $[t]_E \notin [T]_E$ and $E \cup \{t \approx c\} \models e$ then $E \models e$.*

Proof. Assume that $[t]_E \notin [T]_E$ and that $E \cup \{t \approx c\} \models e$. Let E' be a reduced set of rules equivalent to E, such that $c{\downarrow_{E'}} = c$. Let $t' = t{\downarrow_{E'}}$. If $t' = c$ then

$$E \cup \{t \approx c\} \equiv E' \cup \{t \approx c\} \equiv E' \cup \{t' \approx c\} \equiv E$$

and the statement follows immediately. So assume that $t' \neq c$. Let $R = E' \cup \{t' \to c\}$. Let $l \to r$ be a rule in E'. Neither l nor r can be reduced with the rule $t' \to c$ because $[t']_E = [t]_E \notin [T]_E$. Hence R is reduced, and thus canonical [138]. Also, $R \equiv E \cup \{t \approx c\}$. (Note that $t' \in [t]_E$ and $[T]_E = [T]_{E'}$.)

Let $e = t_0 \approx s_0$ and let $u = t_0{\downarrow_R} = s_0{\downarrow_R}$. We have that

$$t_0 \xrightarrow{*}_R u, \quad s_0 \xrightarrow{*}_R u.$$

Consider the reduction $t_0 \xrightarrow{*}_R u$ and let $t_i \longrightarrow t_{i+1}$ be any rewrite step in that reduction. Obviously, if each subterm of t_i is in some congruence class in $[T]_E$ then the rule $t' \to c$ is not applicable since $[t']_E \notin [T]_E$ and it follows also that each subterm of t_{i+1} is in some congruence class in $[T]_E$. It follows by induction on i that the rule $t' \to c$ is not used in the reduction. The same argument holds for $s_0 \xrightarrow{*}_R u$. Hence

$$t_0 \xrightarrow{*}_{E'} u, \quad s_0 \xrightarrow{*}_{E'} u,$$

and thus $E' \models t_0 \approx s_0$. Hence $E \models e$. ⊠

Consider a system S of rigid equations. There is an extreme case of rigid equations that are easy to handle from the point of view of solvability of S, namely the redundant ones:

- A rigid equation is *redundant* if all substitutions solve it.

To decide if a rigid equation $E(x) \vdash_\forall s(x) \approx t(x)$ is redundant, it is enough to decide if $E(c) \models s(c) \approx t(c)$ where c is a new constant.

- The *uniform word problem for ground equations* is the following decision problem. Given a set of ground equations E and a ground equation e, is e a logical consequence of E?

We use the following complexity result [88, 89].

Theorem 6.1 (Kozen) *The uniform word problem for ground equations is P-complete.*

So redundancy of rigid equations is decidable in polynomial time.

Lemma 6.2 *Let $E(x) \vdash_\forall e(x)$ be a rigid equation, c be a new constant and t be a ground term not containing c. Then*

$$E(c) \cup \{t \approx c\} \models e(c) \quad \Leftrightarrow \quad E(t) \models e(t).$$

Proof. The only non-obvious direction is '$\Rightarrow$'. Since t does not include c, $E(c) \cup \{t \approx c\} \models e(c)$ holds with c replaced by t, but then the equation $t \approx t$ is simply superfluous. ⊠

Clearly, S is solvable iff the set of rigid equations in S that are not redundant, is solvable. We use the following lemma.

Lemma 6.3 *Let $E(x) \models_\forall s_0(x) \approx t_0(x)$ be a rigid equation and c be a new constant. There exists a finite set of ground terms T such that, for any ground term t not containing c the following holds:*

$$E(t) \models s_0(t) \approx t_0(t) \quad \Leftrightarrow \quad E(c) \models t \approx s \text{ for some } s \in T.$$

Furthermore, T can be obtained in polynomial time.

Proof. Let T' be the set $Terms(E(c) \cup \{s_0(c) \approx t_0(c)\})$. Let

$$T = \{\, s \in T' \mid E(c) \cup \{s \approx c\} \models s_0(c) \approx t_0(c) \,\}.$$

Note that T may be empty. Let t be any ground term that does not contain c. By using Lemma 6.2, it is enough to prove that the following statements are equivalent:

1. $E(c) \cup \{t \approx c\} \models s_0(c) \approx t_0(c)$,
2. $E(c) \models t \approx s$ for some $s \in T$.

Assume first that $[t]_{E(c)} \notin [T']_{E(c)}$. In particular $[t]_{E(c)} \notin [T]_{E(c)}$, so statement 2 is trivially false. Suppose (by contradiction) that statement 1 holds. But then $E(c) \models s_0(c) \approx t_0(c)$ by Lemma 6.1, which contradicts that the rigid equation is not redundant.

Assume now that $[t]_{E(c)} = [s]_{E(c)}$ for some $s \in T'$. Thus

$$E(c) \cup \{s \approx c\} \equiv E(c) \cup \{t \approx c\}. \tag{6.1}$$

So, if $s \in T$ then statement 2 is trivially true and statement 1 is true by (6.1) and the definition of T. If on the other hand $s \notin T$ then statement 2 is trivially false and statement 1 is false by (6.1) and the definition of T.

Observe that the size of T' is proportional to the size of the rigid equation, and to decide if some term i T' belongs to T takes polynomial time by Kozen's result. So the construction of T takes polynomial time. ⊠

From Lemma 6.3 we get the following result.

Theorem 6.2 *Rigid E-unification with one variable is P-complete.*

Proof. P-hardness of rigid E-unification with one variable follows immediately from P-hardness of the uniform word problem of ground equations. Inclusion in P is proved as follows. Let $S(x) = E(x) \models_\forall e(x)$ be a rigid equation. Test first that $S(x)$ is not redundant. If so, use Lemma 6.3 to obtain T. Now, $S(x)$ is solvable iff T is non-empty. ⊠

This P-completeness result is extended in Section 6.4 to SREU with one variable and a constant bound on the number of rigid equations.

Reduction to Tree Automata

We use the following relationship between tree automata and arbitrary ground rewrite systems [14].

Theorem 6.3 (Brainerd) *Let R be a ground rewrite system and T a finite set of terms. Then the set $\{ t \mid (\exists s \in T)\ t \xrightarrow{*}_R s \}$ is recognizable. Furthermore, a tree automaton that recognizes this set can be obtained effectively from R and T.*

Recently, corresponding connections between recognizability (with respect to pushdown tree automata [129]) and nonground term rewriting systems have been studied by several authors [20, 21, 54, 62, 126]. For a survey of connections between rewriting and tree automata see Dauchet [22]. We obtain the following corollary.

Corollary 6.1 *Let $S(x)$ be a rigid equation with one variable x that is not redundant. Then the set of $x\theta$ such that θ solves $S(x)$ is recognizable. Furthermore, a tree automaton that recognizes this set is obtained effectively from S.*

Proof. Immediate by Brainerd's theorem and Lemma 6.3 ⊠

By using the fact that the class of recognizable sets is (effectively) closed under finite intersectons and that the non-emptiness problem of tree automata is decidable [43, 143], the decidability result of SREU with one variable follows from Corollary 6.1. The decidability is proved formally below, with a precise computational complexity bound.

6.4 COMPUTATIONAL COMPLEXITY

In the previous section we showed that SREU with one variable is decidable. We paid little or no attention to the actual computational complexity of this decision problem. Here we take a closer look at the reduction and show that SREU with one variable is in fact EXPTIME-complete. Recall the following definition.

- The *intersection non-emptiness problem of DTAs* is the following decision problem. Given a collection $\{ A_i \mid 1 \leq i \leq n \}$ of DTAs, is $\bigcap_{i=1}^{n} T(A_i)$ non-empty?

Let us first show that SREU with one variable reduces to the intersection non-emptiness problem of DTAs in polynomial time. This establishes the inclusion of SREU with one variable in EXPTIME. We then show that the intersection non-emptiness problem of DTAs reduces to SREU with one variable, which shows the hardness part. To show the first part we have to be more precise about how a DTA can be constructed from a given set of equations. Brainerd's theorem is too general here and its computational complexity is unclear. Instead of using Brainerd's theorem, we give an explicit construction of a DTA from a given set of equations. This construction is in fact based on a construction in de Kogel [27, Theorems 4.1 and 4.2] that is based on Shostak's congruence closure algorithm [136].[1] A similar construction is used also in Dauchet, Heuillard, Lescanne and Tison [24], and in Gurevich and Voronkov [73].

Inclusion in EXPTIME

In the following we assume that none of the rigid equations are redundant. Lemma 6.3 tells us that the set of solutions of a rigid equation $E(x) \models_{\forall} e(x)$ with one variable is given by the union of a finite number of congruence classes

$$\bigcup_{s \in T} \{ t \mid E(c) \models s \approx t \},$$

where $T \subseteq \mathit{Terms}(E(c) \cup \{e(c)\})$ and c is a new constant. We now give a polynomial time construction of a DTA that recognizes the above set of terms. Our considerations lead naturally to the following definition. Let E be a set of ground equations and T a subset of $\mathit{Terms}(E)$.

- A DTA $A = (Q, \Sigma, R, F)$ is *presented by* (E, T) if A has the following form (modulo renaming of states). First, let q_C be a new state for each $C \in [\mathit{Terms}(E)]_E$.

  $Q \quad = \quad \{ q_C \mid C \in [\mathit{Terms}(E)]_E \},$

[1] De Kogel does not use tree automata but the main idea is the same.

$$\begin{aligned}\Sigma &= \Sigma(E),\\ F &= \{\, q_C \mid C \in [T]_E \,\},\\ R &= \{\, f(q_{[t_1]_E}, \ldots, q_{[t_n]_E}) \to q_{[t]_E} \mid t = f(t_1, \ldots, t_n) \in \mathit{Terms}(E) \,\}.\end{aligned}$$

It is clear that the above definition is well defined. It follows from elementary properties of congruence relations that A is deterministic and thus R is reduced. Note that for each constant c in $\Sigma(E)$, there is a rule $c \to q_{[c]_E}$ in R. Note also that for any equation $s \approx t$ in E, both s and t reduce to the same normal form $q_{[s]_E} = q_{[t]_E}$ with respect to R, since they belong to $\mathit{Terms}(E)$. We use the following lemma.

Lemma 6.4 *Let E be a set of ground equations and $T \subseteq \mathit{Terms}(E)$. Let A be a DTA presented by (E, T). Then*

1. $T(A) = \{\, t \in \mathcal{T}_{\Sigma(E)} \mid (\exists s \in T)\, E \models t \approx s \,\}$,
2. *A can be constructed in polynomial time from E and T.*

Proof. To prove the first statement, consider a Σ-structure $\mathfrak{A}$ with the universe $\{\, t{\downarrow_R} \mid t \in \mathcal{T}_{\Sigma\cup\Gamma} \,\}$ and the interpretation function such that $t^{\mathfrak{A}} = t{\downarrow_R}$ for all $t \in \mathcal{T}_\Sigma$. Clearly, it is enough to prove that, for all $t, s \in \mathcal{T}_\Sigma$,

$$E \models t \approx s \quad \Leftrightarrow \quad \mathfrak{A} \models t \approx s.$$

For a proof of this statement see de Kogel [27].

The second part is proved as follows. The number of terms in $\mathit{Terms}(E)$ is proportional to the size of E. It follows by Theorem 6.1 that the time complexity of the construction of Q, i.e., the time complexity to partition $\mathit{Terms}(E)$ into congruence classes, is polynomial. The rest is obvious. ⊠

We prove now that SREU with one variable is in EXPTIME.

Lemma 6.5 *SREU with one variable is in EXPTIME.*

Proof. Let $S(x) = \{\, S_i(x) \mid 1 \le i \le n \,\}$ be a system of rigid equations. Assume, without loss of generality, that none of the rigid equations is redundant. Let $S_i(x) = E_i(x) \nvDash_\forall e_i(x)$. Let Σ be the signature of S. Use Lemma 6.3 to obtain, for each i, $1 \le i \le n$, a set of ground terms T_i in polynomial time such that, for all t in $\mathcal{T}_\Sigma$,

$$E_i(t) \models e_i(t) \quad \Leftrightarrow \quad E_i(c) \models t \approx s \text{ for some } s \in T_i.$$

Use now Lemma 6.4 to obtain (in polynomial time) a DTA A_i that presents $(E_i(c), T_i)$, for $1 \leq i \leq n$. It follows by Lemma 6.3 and the first part of Lemma 6.4 that

$$T(A_i) = \{\, t \in \mathcal{T}_\Sigma \mid E_i(t) \models e_i(t) \,\} \quad (\text{for } 1 \leq i \leq n).$$

Thus, θ is a solution to $S(x)$ iff $x\theta$ is recognizable by all $T(A_i)$. Consequently, $S(x)$ is solvable iff $\bigcap_{i=1}^{n} T(A_i)$ is non-empty. The lemma follows, since the intersection non-emptiness problem of DTAs is in EXPTIME. ⊠

EXPTIME-completeness

We reduce the intersection non-emptiness problem of DTAs to SREU with one variable to establish the hardness part. First, let us state some simple but useful facts.

Lemma 6.6 *Let $A = (Q, \Sigma, R, F)$ be a DTA, f a unary function symbol not in Σ, and c a constant not in Q or Σ. Let*

$$S(x) = (R \cup \{\, f(q) \to c \mid q \in F \,\} \models_\forall x \approx c).$$

Then, for all θ such that $x\theta \in \mathcal{T}_{\Sigma \cup \{f\}}$,

$$\theta \text{ solves } S(x) \quad \Leftrightarrow \quad x\theta = f(t) \text{ for some } t \in T(A).$$

Proof. Let $E = R \cup \{\, f(q) \to c \mid q \in F \,\}$. From the fact that R is reduced and that $f(q)$ is irreducible in E and c is irreducible in R, follows that E is reduced and thus canonical. So, for any $x\theta \in \mathcal{T}_{\Sigma \cup \{f\}}$, $E \models x\theta \approx c$ iff $x\theta \xrightarrow{*}_E c$. But

$$\begin{aligned} x\theta \xrightarrow{*}_E c \quad &\Leftrightarrow \quad x\theta \xrightarrow{*}_E f(q) \longrightarrow c \text{ for some } q \in F \\ &\Leftrightarrow \quad x\theta = f(t) \text{ for some } t \in \mathcal{T}_\Sigma \text{ and } t \xrightarrow{*}_R q \\ &\Leftrightarrow \quad x\theta = f(t) \text{ for some } t \in T(A). \end{aligned}$$

⊠

For a given signature Σ, and some constant c in it, let us denote by $S_\Sigma(x)$ the following rigid equation:[2]

$$S_\Sigma(x) = (\{\, f(c, \ldots, c) \approx c \mid f \in \Sigma \,\} \models_\forall x \approx c).$$

The following lemma is elementary [38].

Lemma 6.7 *For all θ, θ solves $S_\Sigma(x)$ iff $x\theta \in \mathcal{T}_\Sigma$.*

[2] Note that $f(c, \ldots, c)$ stands for f when f is a constant.

We have now reached the point where we can state and easily prove the following result.

Theorem 6.4 *SREU with one variable is EXPTIME-complete.*

Proof. Inclusion in EXPTIME follows by Lemma 6.5. Let $\{ A_i \mid 1 \leq i \leq n \}$ be a collection of DTAs with signature Σ. Let f be a new unary function symbol and $\Sigma' = \Sigma \cup \{f\}$. For each A_i, let $S_i(x)$ be the rigid equation given by Lemma 6.6. So, for all θ such that $x\theta \in \mathcal{T}_{\Sigma'}$,

$$\theta \text{ solves } S_i(x) \quad \Leftrightarrow \quad x\theta = f(t) \text{ for some } t \in T(A_i).$$

Let

$$S(x) = \{ S_i(x) \mid 1 \leq i \leq n \} \cup \{S_{\Sigma'}(x)\}.$$

It follows by Lemma 6.7 that for any θ that solves $S(x)$, $x\theta$ is in $\mathcal{T}_{\Sigma'}$. Hence, by Lemma 6.6, $S(x)$ is solvable iff $\bigcap_{i=1}^{n} T(A_i)$ is non-empty. Obviously, $S(x)$ has been constructed in polynomial time. The statement follows from Theorem 5.2. ⊠

So in the general case, SREU is already intractable with one variable. It should be noted, however, that the exponential behaviour is strongly related to the unboundedness of the number of rigid equations. (See Section 6.4.)

Bounded Case

The exponential worst case behaviour of SREU with one variable is strongly related to the unboundedness of the number of rigid equations, and not to the size or other parameters of the rigid equations. This behaviour is explained by the fact that the intersection non-emptiness problem of a family of DTAs is in fact the non-emptiness problem of the corresponding direct product of the family. The size of a direct product of a family of DTAs is proportional to the product of the sizes of the members of the family, and the time complexity of the non-emptiness problem of a DTA is polynomial.

- *Bounded SREU* is SREU with a number of rigid equations that is bounded by some fixed positive integer.

For bounded SREU with one variable we get the following result.

Theorem 6.5 *Bounded SREU with one variable is P-complete.*

Proof. Let the number of rigid equations be bounded by some fixed positive integer n. P-hardness follows immediately from Theorem 6.2. Without loss of generality consider a system

$$S(x) = \{ S_i(x) \mid 1 \leq i \leq n \}$$

of exactly n rigid equations. For each S_i construct a DTA A_i in polynomial time, like in Lemma 6.5. Let A be the DTA that recognizes $\bigcap_{i=1}^{n} T(A_i)$. For example, A can be the direct product of $\{ A_i \mid 1 \leq i \leq n \}$ (Gécseg and Steinby [60]). It is straightforward to construct A in time that is proportional to the product of the sizes of the A_i's. Hence A is obtained in polynomial time (because n is fixed) and $T(A)$ is non-empty iff $S(x)$ is solvable. The statement follows from Theorem 5.1. ⊠

Monadic Case

When we restrict the signature to consist of function symbols of arity ≤ 1, i.e., when we consider the so-called *monadic* SREU then the complexity bounds are different. DTAs restricted to signatures with just unary function symbols correspond to classical deterministic finite automata or DFAs. The following result is proved in Kozen [90].

Theorem 6.6 (Kozen) *The intersection non-emptiness problem of DFAs is PSPACE-complete.*

We get the following result.

Theorem 6.7 *Monadic SREU with one variable is PSPACE-complete.*

Proof. Inclusion in PSPACE follows from Lemma 6.5 trivially modified so that Theorem 6.6 is used. PSPACE-hardness follows from Theorem 6.4 trivially modified so that Theorem 6.6 is used. ⊠

A detailed study of monadic SREU can be found elsewhere [73], where also the PSPACE-completeness is proved. We can note that, in general, the decidability of monadic SREU is still an open problem [73].

6.5 UNITED ONE VARIABLE CASE

In this section we extend the decidability result of SREU with one variable to SREU with multiple variables with the following syntactical restriction on the structure of each rigid equation. We say that a system of rigid equations has the *united one variable property* if each rigid equation $E \models_\forall e$ in it satisfies the following conditions:

1. Either $E \not\models e$ includes at most one variable, or
2. E is ground and e has the form $x \approx y$ for two variables x and y.

SREU restricted to systems with the united one variable property is called *united one variable SREU*. This is a nontrivial extension of the one variable case.

Example 6.1 Let $S_{\mathrm{id}}(x)$, $S_{\mathrm{mv}}(y)$ and $S_1(x,y)$ be the rigid equations defined in Chapter 3, then the system

$$\{\, S_{\mathrm{id}}(x),\ S_{\mathrm{mv}}(y),\ S_1(x,y)\,\}$$

has the united one variable property. Note that, by adding the rigid equation $S_2(x,y)$ we violate the united one variable property because $S_2(x,y)$ contains more than one variable and its right-hand side is not a simple equality between two variables. □

The main result of this section is that the united one variable SREU is decidable. The proof is by reduction to the decidable first-order theory of ground rewrite systems [26].

The Decidable Theory GRS

Now we formally define the *theory of ground rewrite systems* or *GRS*. Consider a signature Σ that contains all the function symbols and constants that we are going to need in the sequel. Let Γ be the following signature constructed from Σ.

- For each term t in $\mathcal{T}_\Sigma$, let $\bar{t}$ be a constant in Γ.
- For each ground rewrite system E over $\mathcal{T}_\Sigma$, let R_E be a new binary relation symbol in Γ.[3]

Now, let $\mathfrak{A}$ be the following Γ-structure. The universe of $\mathfrak{A}$ is $\mathcal{T}_\Sigma$ and the interpretation function of $\mathfrak{A}$ is defined as follows. Note that the only ground terms in the signature of $\mathfrak{A}$ are the constants $\bar{t}$ for $t \in \mathcal{T}_\Sigma$, since there are no function symbols in Γ of positive arity.

1. For each constant $\bar{t} \in \Gamma$, $\bar{t}^{\mathfrak{A}} = t$.
2. For each relation symbol $R_E \in \Gamma$, $R_E^{\mathfrak{A}}$ is the rewrite relation $\xrightarrow{*}_E$.

[3] In the original definition of GRS [26] there are two more relation symbols for each E, but we do not use them here.

We can now define GRS as the first-order theory of $\mathfrak{A}$, i.e.,

$$\mathrm{GRS} = \{\, \varphi \text{ a sentence in } \Gamma \mid \mathfrak{A} \models \varphi \,\}.$$

We use the following result [26].

Theorem 6.8 (Dauchet–Tison) *GRS is decidable.*

The proof of Theorem 6.8 is by reduction to finite tree automata. In particular, it involves, for each ground rewrite system, a construction of a "ground tree transducer" that is a pair of a bottom-up and a top-down finite tree automaton, and defines the rewrite relation that is related with that rewrite system [23, 25]. When GRS is restricted to *reduced* ground rewrite systems (which is enough in our case) one can give an easier proof of Theorem 6.8 by reduction to the decidable weak monadic second-order theory of the binary tree or WS2S.[4] See Thomas [144] for a survey of related topics.

Reduction to GRS

We use the following lemma. In the following we consider rigid equations in a fixed signature Σ that contains at least one constant. We also assume that we have a sufficiently large supply of new constants.

Lemma 6.8 *Let $E(x) \models_\forall e(x)$ be a non-redundant rigid equation with one variable x. There is a formula $\varphi(x)$ in the language of GRS such that, for all ground terms t,*

$$\mathfrak{A} \models \varphi(\bar{t}) \quad \Leftrightarrow \quad E(t) \models e(t) \text{ and } t \in \mathcal{T}_\Sigma.$$

Proof. Let c be a new constant and use Lemma 6.3 to obtain a finite set T ($\subseteq \mathcal{T}_{\Sigma\cup\{c\}}$) of ground terms such that, for all ground terms t not containing c,

$$E(t) \models e(t) \quad \Leftrightarrow \quad E(c) \models t \approx s \text{ for some } s \in T.$$

Let $E_\Sigma = \{\, f(c_1,\ldots,c_1) \approx c_1 \mid f \in \Sigma \,\}$[5] where c_1 is some constant in Σ. Consider both $E(c)$ and E_Σ as rewrite systems, with equations as rules in both directions. Let $\varphi(x)$ be the following formula:

$$\varphi(x) = (\bigvee_{s\in T} R_{E(c)}(x,\bar{s})) \wedge R_{E_\Sigma}(x,\bar{c}_1).$$

[4] Such a proof has been given by Gurevich and Veanes.

[5] Note that $f(c_1,\ldots,c_1)$ stands for f whenever f is a constant.

It follows by definition of $\mathfrak{A}$ that, for all ground terms t,

$$\begin{aligned}
\mathfrak{A} \models \varphi(\bar{t}) &\Leftrightarrow \mathfrak{A} \models \bigvee_{s \in T} R_{E(c)}(\bar{t}, \bar{s}) \text{ and } \mathfrak{A} \models R_{E_\Sigma}(\bar{t}, \bar{c}_1) \\
&\Leftrightarrow t \xrightarrow{*}_{E(c)} s \text{ for some } s \in T, \text{ and } t \xrightarrow{*}_{E_\Sigma} c_1 \\
&\Leftrightarrow E(c) \models t \approx s \text{ for some } s \in T, \text{ and } t \in \mathcal{T}_\Sigma \\
&\Leftrightarrow E(t) \models e(t) \text{ and } t \in \mathcal{T}_\Sigma,
\end{aligned}$$

where the last equivalence holds by the above, because c is not in Σ. ⊠

We can now prove the following.

Theorem 6.9 *United one variable SREU is decidable.*

Proof. Let $S = \{ S_i \mid 1 \leq i \leq n \}$ be a system of rigid equations with the united one variable property. Assume, without loss of generality, that none of the rigid equations in S is redundant. For each rigid equation $S_i(x)$ in S with one variable x let $\varphi_i(x)$ be the formula given by Lemma 6.8. For each rigid equation $S_i(x, y) = E_i \nvdash_\forall x \approx y$ in S, where E_i is ground, and x and y are variables, consider E_i as a ground rewrite system with equations as rules in both directions and let $\varphi_i(x, y) = R_{E_j}(x, y)$. So, for all ground terms t and s,

$$E_i \models t \approx s \quad \Leftrightarrow \quad t \xrightarrow{*}_{E_i} s \quad \Leftrightarrow \quad \mathfrak{A} \models R_{E_i}(\bar{t}, \bar{s}).$$

Finally, let φ be the existential closure of the conjunction of all the φ_i's. It is straightforward to verify that φ is a theorem in GRS iff S is solvable. The statement follows by Theorem 6.8. ⊠

The computational complexity of the united one variable SREU is not known, we know only that it is at least EXPTIME-hard. It also remains to be investigated if there are other decidable extensions of the one variable case. We can also note the following result. The $\exists$-*fragment* of GRS is the set of prenex formulas in GRS with one existential quantifier.

Corollary 6.2 *The $\exists$-fragment of GRS is EXPTIME-hard.*

Proof. From the proof of Theorem 6.9 it is clear that the reduction from SREU with one variable to GRS can be performed in polynomial time and that the resulting formula is a prenex formula with one existential quantifier. The statement follows now from Theorem 6.4. ⊠

Prenex Fragment of Intuitionistic Logic

7.1 INTRODUCTION

The strong connections between SREU and intuitionistic logic with equality have implied new important decidability results in the latter area [35, 155]. From the undecidability of SREU follows, for example, that the $\exists^*$-fragment of intuitionistic logic with equality is undecidable [37, 38]. By using the results in Chapter 3, we improve this result to the following.

> *The $\exists\exists$-fragment of intuitionistic logic with equality is undecidable.*

The decidability of the $\forall^*\exists\forall^*$-fragment of intuitionistic logic with equality has been an open problem which is settled in this chapter by using the results in Chapter 6 and an analogue of a Skolemization result for intuitionistic logic [35]. The following is proved.

> *The $\forall^*\exists\forall^*$-fragment of intuitionistic logic with equality is decidable and EXPTIME-hard.*

The above two results imply the following main contribution of this chapter. The *prenex fragment* of intuitionistic logic is the collection of all intuitionistically provable prenex formulas.

> *A complete classification of decidability of the prenex fragment of intuitionistic logic with equality, in terms of the quantifier prefix.*

At the end of this chapter we compare these fragments with the corresponding fragments in classical logic.

$$\frac{}{\Gamma, A, \Delta \to A}\ (Ax) \qquad \frac{}{\Gamma, \bot, \Delta \to \varphi}\ (\bot) \qquad \frac{}{\Gamma \to t \approx t}\ (\approx)$$

Axioms

$$\frac{\Gamma\{s/x\}, s \approx t, \Delta\{s/x\} \to \chi\{s/x\}}{\Gamma\{t/x\}, s \approx t, \Delta\{t/x\} \to \chi\{t/x\}}\ (\approx_1) \qquad \frac{\Gamma\{s/x\}, t \approx s, \Delta\{s/x\} \to \chi\{s/x\}}{\Gamma\{t/x\}, t \approx s, \Delta\{t/x\} \to \chi\{t/x\}}\ (\approx_2)$$

Replacement rules

$$\frac{\Gamma, \varphi, \psi, \Delta \to \chi}{\Gamma, \varphi \wedge \psi, \Delta \to \chi}\ (\wedge \to) \qquad \frac{\Gamma \to \varphi \quad \Gamma \to \psi}{\Gamma \to \varphi \wedge \psi}\ (\to \wedge)$$

$$\frac{\Gamma, \varphi, \Delta \to \chi \quad \Gamma, \psi, \Delta \to \chi}{\Gamma, \varphi \vee \psi, \Delta \to \chi}\ (\vee \to) \qquad \frac{\Gamma \to \varphi}{\Gamma \to \varphi \vee \psi}\ (\to \vee_1) \quad \frac{\Gamma \to \psi}{\Gamma \to \varphi \vee \psi}\ (\to \vee_2)$$

$$\frac{\Gamma, \psi, \Delta \to \chi \quad \Gamma, \varphi \Rightarrow \psi, \Delta \to \varphi}{\Gamma, \varphi \Rightarrow \psi, \Delta \to \chi}\ (\Rightarrow\to) \qquad \frac{\varphi, \Gamma \to \psi}{\Gamma \to \varphi \Rightarrow \psi}\ (\to\Rightarrow)$$

Propositional rules

$$\frac{\Gamma, \varphi\{t/x\}, \forall x\varphi, \Delta \to \chi}{\Gamma, \forall x\varphi, \Delta \to \chi}\ (\forall \to) \qquad \frac{\Gamma \to \varphi\{y/x\}}{\Gamma \to \forall x\varphi}\ (\to \forall)$$

$$\frac{\Gamma, \varphi\{y/x\}, \Delta \to \chi}{\Gamma, \exists x\varphi, \Delta \to \chi}\ (\exists \to) \qquad \frac{\Gamma \to \varphi\{t/x\}}{\Gamma \to \exists x\varphi}\ (\to \exists)$$

Quantifier rules

Figure 7.1: Inference rules of $\mathrm{LJ}^{\approx}$. Here A stands for an atomic formula; φ, ψ and χ stand for arbitrary formulas; Γ and Δ stand for multisets of formulas. In the rules $(\exists \to)$ and $(\to \forall)$ the variable y does not occur free in the conclusions of the rules.

7.2 PRELIMINARIES

We use $\vdash_{\mathrm{c}}$ for classical provability and $\vdash_{\mathrm{i}}$ for intuitionistic provability. The particular choice of formal system does not affect the decidability results. For example, we can assume natural deduction for $\vdash_{\mathrm{c}}$ and natural deduction without the RAA rule for $\vdash_{\mathrm{i}}$. (See for example Troelstra and van Dalen [145].)

A *sequent* is an expression of the form $\Delta \to \varphi$, where Δ is a multiset of formulas and φ a formula. In Section 7.3, we consider derivations in a particular (cut-free) sequent calculus $\mathrm{LJ}^{\approx}$ for intuitionistic logic with equality. (See Figure 7.1.) Any negated formula $\neg\varphi$ is a shorthand for $\varphi \Rightarrow \bot$, where $\bot$ is a propositional constant. It is well-known that a closed sequent $\to \varphi$ is derivable in $\mathrm{LJ}^{\approx}$ iff $\vdash_{\mathrm{i}} \varphi$, see, e.g., Orevkov [110].

We use the following general properties.

- The *explicit definability* property of intuitionistic logic: if $\vdash_{\mathrm{i}} \exists x\varphi$ then there is a ground substitution θ such that $\vdash_{\mathrm{i}} \varphi\theta$.

- If φ is a conjunction of closed implications of the form $\psi \Rightarrow e$, where ψ is a conjuntion of equations and e is an equation, then $\vdash_{\mathrm{c}} \varphi$ iff $\vdash_{\mathrm{i}} \varphi$.

A *prenex formula* is a formula of the form

$$Q_1 x_1 \ldots Q_n x_n \varphi(x_1, \ldots, x_n)$$

where φ is quantifier free and each Q_i is a quantifier. The *prenex fragment* of intuitionistic logic is the class of all intuitionistically provable prenex formulas. For a rigid equation $E \fatslash e$ let

$$\mathcal{F}(E \fatslash e) \quad = \quad (\bigwedge_{d \in E} d) \Rightarrow e,$$

and for a system S of rigid equations let

$$\mathcal{F}(S) \quad = \quad \bigwedge \{\, \mathcal{F}(S_1) \mid S_1 \in S \,\}.$$

7.3 CLASSIFICATION OF THE PRENEX FRAGMENT

The decidability problem of the fragments of intuitionistic logic has not been as thoroughly studied as the corresponding problem in classical logic, where the decidability of all standard fragments has been systematically classified [13]. In particular, not much has been known about the prenex fragment (of intuitionistic logic). Many new decidability results about the prenex fragment have been obtained quite recently by Degtyarev and Voronkov [34, 35, 37, 38] and Voronkov [153, 154]. Some of these results are:

1. Decidability, and in particular PSPACE-completeness, of the prenex fragment of intuitionistic logic *without* equality [153, 154].
2. Prenex fragment of intuitionistic logic *with* equality but *without* function symbols is PSPACE-complete [35]. Decidability of this fragment was proved in Orevkov [111].
3. Prenex fragment of intuitionistic logic with equality in the language with one unary function symbol is decidable [35].
4. $\exists^*$-fragment of intuitionistic logic with equality is undecidable [34, 37, 38].

In many of the above results, the corresponding result has first been obtained for a fragment of SREU with similar restrictions. In particular, the last statement follows from the undecidability of SREU and the following property, originally used in Degtyarev and Voronkov [34, Theorem 4].

Lemma 7.1 *Let $S(\vec{x})$ be a system of rigid equations.*

$$S \text{ is solvable} \quad \Leftrightarrow \quad \vdash_{\mathrm{i}} \exists \vec{x} \mathcal{F}(S).$$

Proof. The direction '$\Rightarrow$' follows by the fact that if there exists a θ that solves S then $\vdash_{\mathrm{c}} \mathcal{F}(S)\theta$ and thus $\vdash_{\mathrm{i}} \mathcal{F}(S)\theta$ (for this class of formulas). Hence $\vdash_{\mathrm{i}} \exists \vec{x} \mathcal{F}(S)$. The direction '$\Leftarrow$' follows by the explicit definability property of intuitionistic logic, which provides a solution for S. ⊠

The $\exists\exists$-Fragment

By using some results from Chapter 3, we obtain a uniform characterization of all the recursively enumerable sets in the $\exists\exists$-fragment of intuitionistic logic with equality. Let us consider Turing machines with some fixed tape alphabet and a fixed symbol q_0 for the initial state. Let t_v denote the word that represents $q_0 v$ (the initial ID for input string v).

Recall the following: given a TM M we can construct a system $\hat{S}_v^M(x, y)$ of three rigid equations such that the left-hand sides are ground and independent of v, and $\hat{S}_v^M(x, y)$ is solvable iff M accepts v.

Theorem 7.1 *Uniformly, for any Turing machine M, there is a formula*

$$\varphi^M(z, x, y) \quad = \quad (E \Rightarrow x \,.\, y \approx c) \wedge (\Pi_1 \Rightarrow x \approx y) \wedge (\Pi_2 \Rightarrow x \approx z \,.\, y).$$

where E, Π_1 and Π_2 are closed conjunctions of equations and c is a constant, such that, for all input strings v for M,

$$\vdash_i \exists x \exists y \varphi^M(t_v, x, y) \quad \Leftrightarrow \quad M \text{ accepts } v.$$

Proof. Let $\varphi^M(t_v, x, y) = \mathcal{F}(\hat{S}_v^M)$. Use Lemma 7.1 and Theorem 3.3. ⊠

Let us now consider a universal Turing machine M_{u} and let $\varphi^{\mathrm{u}} = \varphi^{M_{\mathrm{u}}}$ be the formula given by Theorem 7.1. Let us also write $\varphi_v^M(x, y)$ for $\varphi^M(t_v, x, y)$. We get the following result.

Theorem 7.2 *The $\exists\exists$-fragment of intuitionistic logic is undecidable already under the following restrictions:*

1. *The signature has two symbols: one constant and one binary function symbol.*
2. *The only connectives are $\wedge$ and at most three $\Rightarrow$'s.*
3. *The antecedents of all implications are closed.*

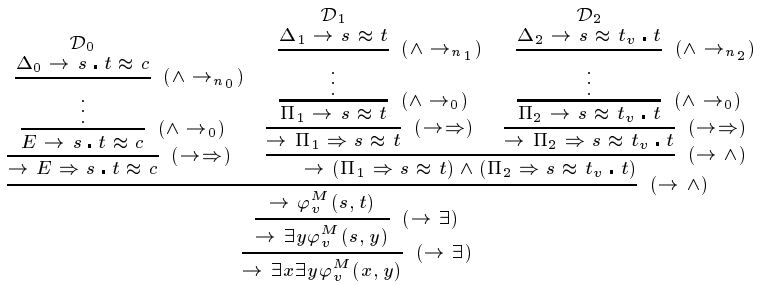


Figure 7.2: A derivation of $\exists x \exists y \varphi_v^M(x,y)$ in $\mathrm{LJ}^{\approx}$; Δ_0, Δ_1 and Δ_2 are multiset of equations corresponding to E, Π_1 and Π_2, respectively; n_0, n_1 and n_2 are the number of $\wedge$'s minus one, in E, Π_1 and Π_2, respectively. It is actually the existence of the derivations $\mathcal{D}_0$, $\mathcal{D}_1$ and $\mathcal{D}_2$, that corresponds to the solvability problem of the system S_v^M of rigid equations.

4. The antecedents of implications may be fixed.

Proof. Let M be a TM and v an input string for M. Then, by Theorem 7.1,

$$\vdash_{\mathrm{i}} \exists x \exists y \varphi^{\mathrm{u}}_{\langle M,v\rangle}(x,y) \quad \Leftrightarrow \quad M_u \text{ accepts } \langle M, v\rangle \quad \Leftrightarrow \quad M \text{ accepts } v.$$

Furthermore, all constants in $\varphi^{\mathrm{u}}_{\langle M,v\rangle}$ can be simulated by one constant and one binary function symbol (see the remark following Corollary 3.3). ⊠

A Remark on Skeleton Instantiation

Proof search in intuitionistic logic with equality is closely connected with SREU, and, unlike in the classical case, the handling of SREU is in fact *unavoidable* in that context [152, 155]. A *skeleton* is the structure of a derivation in $\mathrm{LJ}^{\approx}$, where all the replacement rules have been removed. *Skeleton instantiation* is the problem of the existence of a derivation with a given skeleton. The structure of a skeleton is given, to a certain extent, by the structure of the formula that one searches a proof for. Voronkov shows that SREU is polynomially equivalent to skeleton instantiation in the sequent calculus $\mathrm{LJ}^{\approx}$ [152].

The sequent $\to \exists x \exists y \varphi_v^M(x,y)$ has a derivation in $\mathrm{LJ}^{\approx}$ iff it has a derivation in $\mathrm{LJ}^{\approx}$ of the form shown in Figure 7.2 where the derivations $\mathcal{D}_i$ consist solely of replacement rules and the rule $(\approx)$ at the top. The skeleton of the derivation in Figure 7.2 is illustrated in Figure 7.3. We can draw the following conclusion from this.

Corollary 7.1 *There is a fixed skeleton with two applications of* $(\to \exists)$ *and three applications of* $(\to\Rightarrow)$ *for which the skeleton instantiation problem in* $\mathrm{LJ}^{\approx}$ *is undecidable.*

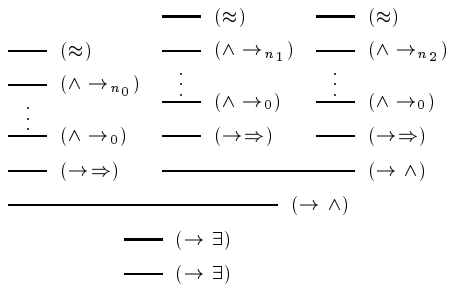


Figure 7.3: The skeleton of the derivation in Figure 7.2.

Proof. By using the results proved in Voronkov [152, 155], the sentence $\exists x \exists y \varphi_v^M(x,y)$ is intuitionistically provable iff the sequent $\rightarrow \exists x \exists y \varphi_v^M(x,y)$ can be derived in $\mathrm{LJ}^{\approx}$ with the skeleton shown in Figure 7.3. Let $M = M_{\mathrm{u}}$. The statement follows now from Theorem 7.1. ⊠

The $\forall^*\exists\forall^*$-Fragment

With the following result we obtain a complete classification of decidability of the prenex fragment of intuitionistic logic with equality, in terms of the quantifier prefix.

Theorem 7.3 *The $\forall^*\exists\forall^*$-fragment of intuitionistic logic with equality is decidable and EXPTIME-hard.*

Proof. We prove the decidability first. Intuitionistic provability of any prenex sentence with the prefix $\forall^*\exists\forall^*$ can be reduced to provability of a prenex sentence with prefix $\exists$ by using an analogue of Skolemization for intuitionistic logic [35, Theorem 3.2]. So consider a sentence $\exists x \varphi(x)$, where φ is quantifier free. The sequent $\rightarrow \exists x \varphi(x)$ is derivable in $\mathrm{LJ}^{\approx}$ iff it is derivable with some skeleton $\mathcal{S}$ and only finitely many skeletons need to be considered and can be computed from φ [156].[1] From φ and a skeleton $\mathcal{S}$ one obtains a system $S(x)$ of rigid equations with one variable x such that $S(x)$ is solvable iff $\rightarrow \exists x \varphi(x)$ is derivable with skeleton $\mathcal{S}$ [156]. The decidability follows now from Theorem 6.4.

Conversely, SREU with one variable reduces in polynomial time, by Lemma 7.1, to intuitionistic provability of a closed $\exists$-formula. So the latter problem is EXPTIME-hard by Theorem 6.4. ⊠

[1] The use of the rule $(\Rightarrow\rightarrow)$ can be restricted so that the premises are guaranteed to become "smaller" than the conclusion.

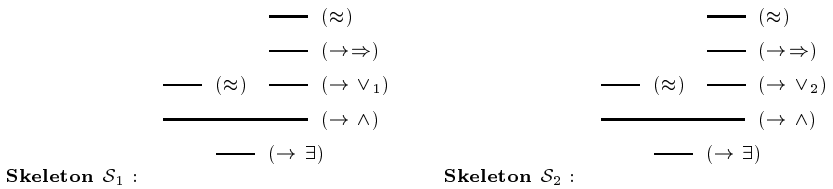


Figure 7.4: The only two possible skeletons in $\mathrm{LJ}^{\approx}$ for the sequent $\to \exists x(x \approx c) \wedge ((x \approx 0 \Rightarrow 0 \approx 1) \vee (x \approx 1 \Rightarrow 0 \approx 1))$.

$$\dfrac{\dfrac{\mathcal{D}_1}{\to x\theta \approx c} \quad \dfrac{\dfrac{\mathcal{D}_2}{x\theta \approx 0 \to 0 \approx 1}\,(\to\Rightarrow)}{\to x\theta \approx 0 \Rightarrow 0 \approx 1}}{\dfrac{\to (x\theta \approx c) \wedge ((x\theta \approx 0 \Rightarrow 0 \approx 1) \vee (x\theta \approx 1 \Rightarrow 0 \approx 1))}{\to \exists x(x \approx c) \wedge ((x \approx 0 \Rightarrow 0 \approx 1) \vee (x \approx 1 \Rightarrow 0 \approx 1))}\,(\to \exists)}\,(\to \wedge)}\,(\to \vee_1)$$

Figure 7.5: Any derivation of the sequent

$$\to \exists x(x \approx c) \wedge ((x \approx 0 \Rightarrow 0 \approx 1) \vee (x \approx 1 \Rightarrow 0 \approx 1))$$

in $\mathrm{LJ}^{\approx}$ with the skeleton $\mathcal{S}_1$ in Figure 7.4 must have this form for some ground substitution θ.

The reduction in the proof of Theorem 7.3 from an $\exists$-formula to SREU with one variable may take exponential time, so the precise upper bound of the computational complexity for the $\forall^*\exists\forall^*$-fragment is currently unknown. A similar decidability proof of another fragment is given in Degtyarev and Voronkov [35, Theorem 7.1] where the authors first prove the equivalence (when restricted to prenex formulas) of $\mathrm{LJ}^{\approx}$ with another system that is based on Hudelmaier's calculus LG [77] and use properties of the latter system.

Example 7.1 Let c, 0 and 1 be constants and consider the formula

$$\varphi(x) \quad = \quad (x \approx c) \wedge ((x \approx 0 \Rightarrow 0 \approx 1) \vee (x \approx 1 \Rightarrow 0 \approx 1)).$$

The sequent $\to \exists x\varphi(x)$ is derivable in $\mathrm{LJ}^{\approx}$ iff it is derivable with one of the two skeletons shown in Figure 7.4. The derivation of $\to \exists x\varphi(x)$ in $\mathrm{LJ}^{\approx}$ with skeleton $\mathcal{S}_1$ must have the form shown in Figure 7.5, where the derivations $\mathcal{D}_1$ and $\mathcal{D}_2$ exist iff the system

$$\{\models_\forall x \approx c, \quad \{x \approx 0\} \models_\forall 0 \approx 1\}$$

of rigid equations is solvable. It follows in a similar way, that there exists a derivation of $\to \exists x\varphi(x)$ in $\mathrm{LJ}^{\approx}$ with skeleton $\mathcal{S}_2$ iff the system

$$\{\models_\forall x \approx c, \quad \{x \approx 1\} \models_\forall 0 \approx 1\}$$

of rigid equations is solvable. Since neither of the systems is solvable, we conclude that $\not\vdash_{\mathrm{i}} \exists x\varphi(x)$.

On the other hand, if we consider the *formula instantiation* problem or the *1-Skeleton* problem of $\varphi(x)$ we see that, classically, $\varphi(x)$ is equivalent to the formula

$$(x \approx c) \wedge (x \not\approx 0 \vee 0 \approx 1 \vee x \not\approx 1 \vee 0 \approx 1).$$

It is easy to see that the formula instantiation problem of $\varphi(x)$ reduces to the solvability of the following system of rigid equations

$$\{\models_{\forall} x \approx c, \quad \{x \approx 0,\ x \approx 1\} \models_{\forall} 0 \approx 1\}$$

that is clearly solvable with a substitution θ such that $x\theta = c$. □

Note that, although both formula instantiation and intuitionistic provability of existential formulas reduce to SREU, these reductions are fundamentally different, as is illustrated with the above example.

7.4 OTHER FRAGMENTS

Decidability problems for other fragments of intuitionistic logic have been studied by Orevkov [109, 111], Mints [104], Statman [139] and Lifschitz [93]. Orevkov proves that the $\neg\neg\forall\exists$-fragment of intuitionistic logic with function symbols is undecidable [109]. Orevkov classifies the decidability of some other *pseudo-prenex* fragments of intuitionistic logic with equality, i.e., classes of formulas with a prenex that is a string in $\{\exists, \forall, \neg\neg\}^*$ [111]. Lifschitz proves that intuitionistic logic with equality and without function symbols is undecidable, i.e., that the pure constructive theory of equality is undecidable [93]. Statman proves that the intuitionistic propositional logic is PSPACE-complete [139].

7.5 CORRESPONDING CLASSICAL FRAGMENTS

The study of the classical decision problem was initiated by Hilbert at the beginning of this century. The classical decision problem can be formulated as the *provability* or *validity* problem in classical logic:

Given a sentence φ, is φ valid?

Classically, a formula is valid iff its negation is unsatisfiable, so the corresponding *satisfiability* problem is an equivalent formulation of the classical decision problem. Gödel's Incompleteness Theorem [63] was an important breakthrough in logic that implied the undecidability of the classical decision problem in general. The identification of which fragments of the class

of all first-order formulas are decidable had been started already before Gödel's result, and several fragments had been shown to be as hard as the whole problem. By now the classification of all the traditional fragments has been completed [13]. See Gurevich [71] for a popular introduction into the subject.

Classically, all formulas are equivalent to prenex formulas. Formulas in prenex form are traditionally classified by imposing restrictions on the quantifier prefix, the signature and either allowing equality or not. Let us adopt the following notation for classes of formulas [13]. Let $[\mathcal{Q}, \alpha, \beta]$ and $[\mathcal{Q}, \alpha, \beta]_{\approx}$ stand for collections of closed first-order prenex formulas with and without equality, respectively, where

1. $\mathcal{Q}$ is a string over $\{\exists, \forall, \exists^*, \forall^*\}$, indicating that prenex sentences with quantifier prefix $\mathcal{Q}$ are allowed. When all prefixes are allowed then *all* is written for $\mathcal{Q}$.
2. $\alpha = (\alpha_1, \ldots, \alpha_m)$, where each α_i is either a natural number or the first infinite ordinal ω, and indicates that there are α_i *relation symbols* of arity i in the signature. If any number of relation symbols of all arities are allowed then *all* is written for α. If there are no relation symbols then (0) is written for α.
3. β is like α but for *function symbols*.

Note that constants are not allowed with this classification. When considering provability (either classical or intuitionistic), constants behave just like universally quantified variables, so any constant can simply be replaced by a new universally quantified variable. For a class $\mathcal{C}$ of sentences, let us write $\mathcal{C}^{(\vdash_{\mathrm{i}})}$ and $\mathcal{C}^{(\vdash_{\mathrm{c}})}$, for the following fragments:

$$\begin{aligned} \mathcal{C}^{(\vdash_{\mathrm{i}})} &= \{\, \varphi \in \mathcal{C} \mid \vdash_{\mathrm{i}} \varphi \,\}, \\ \mathcal{C}^{(\vdash_{\mathrm{c}})} &= \{\, \varphi \in \mathcal{C} \mid \vdash_{\mathrm{c}} \varphi \,\}. \end{aligned}$$

So, we have that

- $[\forall\exists\exists, (0), (0,1)]_{\approx}^{(\vdash_{\mathrm{i}})}$ is undecidable by Theorem 7.2, and
- $[\forall^*\exists\forall^*, \mathit{all}, \mathit{all}]_{\approx}^{(\vdash_{\mathrm{i}})}$ is decidable by Theorem 7.3.

Note that this notion of classification leaves open the decidability of intuitionistic provability of existential prenex sentences *without constants*, i.e., the fragment $[\exists^*, \mathit{all}, \mathit{all}]_{\approx}^{(\vdash_{\mathrm{i}})}$. Classically, without restrictions on the signature, maximal decidable fragments are

- $[\forall^*, \mathit{all}, \mathit{all}]_{\approx}^{(\vdash_{\mathrm{c}})}$ if equality is allowed [70], and

- $[\forall^*\exists\forall^*, all, all]^{(\vdash_c)}$ if equality is disallowed [69, 98].

Let *CLAUSE* be the class of closed prenex formulas whose quantifier free part is a clause, i.e., a disjunction of literals.[2] In the presence of equality, we have that, the Gurevich fragments

- $([\exists, (0), (0,1)]_\approx \cap \mathit{CLAUSE})^{(\vdash_c)}$ and
- $([\exists, (0), (2)]_\approx \cap \mathit{CLAUSE})^{(\vdash_c)}$ are undecidable [70].

Let *HornCLAUSE* be the class of prenex formulas whose quantifier free part is a (strict) Horn clause, i.e., a clause with exactly one positive literal. The Gurevich fragments are already undecidable when restricted to *HornCLAUSE* [160]. In other words, validity is undecidable for closed implications of the form $\exists x(E \Rightarrow e)$, where E is a conjunction of equations and e is an equation. On the other hand, we know that intuitionistic provability is decidable for such implications, e.g., by Lemma 7.1 and the decidability of rigid E-unification [57].

As an exception to the general rule that intuitionistic fragments of prenex formulas seem to be "easier" than the corresponding classical fragments, we have the following case. Consider prenex formulas whose quantifier free part is a conjunction of Horn clauses with *ground negative literals*, such formulas form a subclass of "ground-negative" formulas, for which classical provability is decidable [157]. However, as we have shown, intuitionistic provability is *undecidable* for prenex formulas whose quantifier free part is a corresponding conjunction of implications.

Note that, without equality,

- $[all, all, all]^{(\vdash_i)}$ is decidable and in fact PSPACE-complete [153],

whereas classically, there are already 11 different minimal standard fragments of first-order logic without equality which are undecidable (9 of which use no function symbols at all) [13]. Note also that

- $[all, all, (0)]_\approx^{(\vdash_i)}$ is decidable [111] and in fact PSPACE-complete [35].

In the presence of equality and only one unary function symbol, the maximal decidable standard fragments of classical logic are,

- the Rabin fragment $[all, (\omega), (1)]_\approx^{(\vdash_c)}$ [121], and

[2] Classically, the corresponding class for satisfiability is the class of *Herbrand formulas*, i.e., prenex formulas whose quantifier free part is a conjunction of literals.

- the Shelah fragment $[\forall^*\exists\forall^*, all, (1)]^{(\vdash_c)}_{\approx}$ [135].

In intuitionistic logic we have that

- $[all, all, (1)]^{(\vdash_i)}_{\approx}$ is decidable [35].

7.6 OPEN CASES

We conclude with the following two open problems regarding intuitionistic provability of closed prenex formulas. Recall that *monadic* SREU is SREU restricted to a signature with function symbols of arity ≤ 1. The decidability of the following two fragments is open:

? $[all, all, (\omega)]^{(\vdash_i)}_{\approx}$ and

? $[\exists^*, all, all]^{(\vdash_i)}_{\approx}$.

The fragment $[all, all, (\omega)]^{(\vdash_i)}_{\approx}$ is decidable if and only if monadic SREU is decidable [35]. Let us write SREU_2 for monadic SREU with two unary function symbols. Then monadic SREU is decidable iff SREU_2 is decidable [32]. If SREU_2 is decidable then there hardly exists a simple proof of that. A fact to support this statement is that the *word equation problem* or *unification under associativity* has a simple reduction to SREU_2 [32]. The word equation problem is a hard combinatorial problem that was proved decidable by Makanin [96]. No interesting upper bounds for computational complexity of the word equation problem are known yet. The monadic SREU is treated in detail in Gurevich and Voronkov [73].

Conclusion

8.1 MAIN CONTRIBUTIONS

The main purpose of the thesis is to gain deeper understanding of SREU. The fundamental role of SREU in several areas of computer science has been shown in numerous results by Degtyarev and Voronkov and others. During the course of this work, it turned out that already very small fragments of SREU can be used in a straightforward manner to express rich mathematical constructions. In particular, SREU with ground left-hand sides can be used to express various problems concerned with finite tree automata. The first main result using such observations, in combination with Plaisted's shifted pairing technique [116] is:

- The minimal known undecidable fragment of SREU (see Theorem 3.3).

A useful tool in proving that result, is the *Train Theorem* (Theorem 3.1). We believe that the Train Theorem is of independent interest.

The Herbrand Skeleton problem of fixed multiplicity m, or the m-Skeleton problem, is of fundamental importance in automated theorem proving methods based on the Herbrand theorem. The m-Skeleton problem was proved undecidable by Voda and Komara [151] by a very complicated argument, shortly after the result of Degtyarev and Voronkov [34]. Contrary to their claim [151], we show that the undecidability of the m-Skeleton problem follows directly from the undecidability of SREU. Moreover, by using Theorem 3.3 we are able to identify:

- The minimal known fragment of classical logic for which the m-Skeleton problem is undecidable, for any given m (see Corollary 4.3).

The main tools that we use to get this this result are the notion of *guardedness* of Horn formulas and the *Partisan Corroboration Theorem* (Theorem 4.1). We believe that the Partisan Corroboration Theorem is of independent interest in logic.

After it had become clear that SREU is undecidable with two variables, we soon realized that SREU with one variable is decidable and reduces to the intersection non-emptiness problem of finite tree automata. The complexity of the latter problem was unclear and had to be proven first. As it turned out, the computational complexity results of the basic decision problems of finite tree automata in general, have not been properly addressed in the literature and we did a limited survey on that subject [149]. This survey is summarized in Table 5.1. We formally proved that:

- The intersection non-emptiness problem of finite tree automata is EXPTIME-complete (see Theorem 5.2).
- The non-emptiness problem is P-complete (see Theorem 5.1).

We also drew some general conclusions relating complexity results of classical finite automata to the corresponding results of finite tree automata. In particular, it seems that if a decision problem for (deterministic) finite automata is complete for a certain space complexity class then the same decision problem for (deterministic) finite tree automata is complete for the corresponding alternating space complexity class, but alternating space is precisely deterministic time, only one exponential higher [17].

Using the above complexity results we were then able to prove that:

- SREU with one variable is EXPTIME-complete (see Theorem 6.4).
- SREU with one variable and a constant bound on the number of rigid equations is P-complete (see Theorem 6.5).

Hence, the intractability of SREU with one variable is strongly related to the *number* of rigid equations. In addition, by using a result of Dauchet and Tison [26], we were able to extend the decidability result of SREU with one variable in a non-trivial way: (see Theorem 6.9)

- SREU is decidable if restricted to rigid equations $E \models_\forall e$ such that
  - $E \models_\forall e$ contains at most one variable, or
  - E is ground and e has the form $x \approx y$ for two variables x and y.

Finally, using the undecidability result of SREU with two variables and the decidability result of SREU with one variable, combined with techniques developed in Degtyarev and Voronkov [35] and Voronkov [156], we got a new result in intuitionistic logic with equality:

- A complete classification of the prenex fragment of intuitionistic logic with equality, in terms of the quantifier prefix:

- The $\exists\exists$-fragment is undecidable (see Theorem 7.2).
- The $\forall^*\exists\forall^*$-fragment is decidable and EXPTIME-hard (see Theorem 7.3).

We show also, improving a result in Voronkov [155], that the skeleton instantiation problem in the context of proof search in intuitionistic logic with equality is already undecidable for some *fixed* skeleton. In fact, such a skeleton is illustrated in Figure 7.3.

8.2 CURRENT STATUS OF SREU

Let us briefly summarize the current status of SREU and the results that have been proven about it. The first decidability proof of rigid E-unification is given in Gallier, Narendran, Plaisted and Snyder [56]. Recently a simpler proof, without computational complexity considerations, has been given by de Kogel [27, 28]. Rigid E-unification is studied also in Choi [19]. We start with the **solved cases**:

- Rigid E-unification with ground left-hand side is NP-complete [91]. Rigid E-unification in general is NP-complete and there exist finite complete sets of unifiers [52, 56].
- Rigid E-unification with one variable is P-complete [29] (see Theorem 6.2). Or, more generally, SREU with one variable and a bounded number of rigid equations is P-complete [29] (see Theorem 6.5).
- If all function symbols have arity ≤ 1 (the *monadic* case) then it follows that SREU is PSPACE-hard [66]. If only one unary function symbol is allowed then the problem is decidable [31, 32]. If only constants are allowed then the problem is NP-complete [32] if there are at least two constants.
- About the monadic case it is known that if there are more than 1 unary function symbols then SREU is decidable iff it is decidable with just 2 unary function symbols [32].
- If the left-hand sides are ground then the monadic case is decidable [73]. Monadic SREU with one variable is PSPACE-complete [73] (see Theorem 6.7).
- The word equation solving [96] (i.e., unification under associativity), which is an extremely hard problem with no interesting known computational complexity bounds, can be reduced to monadic SREU [31].
- Monadic SREU is equivalent to a non-trivial extension of word equations [73].

- Monadic SREU is equivalent to the decidability problem of the prenex fragment of intuitionistic logic with equality with function symbols of arity ≤ 1 [35].
- In general SREU is undecidable [34]. Moreover, SREU is undecidable under the following restrictions:
  - The left-hand sides of the rigid equations are ground [116].
  - Furthermore, there are only two variables [147, 148, 150] and three rigid equations with fixed ground left-hand sides [72] (see Theorem 3.4).
- SREU with one variable is decidable, in fact EXPTIME-complete [29] (see Theorem 6.4). Moreover, united one variable SREU, i.e., SREU restricted to rigid equations that either contain one variable, or have a ground left-hand side and a right-hand side that is a simple equality between two variables, is decidable (see Theorem 6.9).

Note also that SREU is decidable when there are no variables, since each rigid equation can be decided for example by using any congruence closure algorithm or ground term rewriting technique. Actually, the problem is then P-complete because the uniform word problem for ground equations is P-complete [89]. The **unsolved cases** are:

? Decidability of monadic SREU [73].

? Decidability of SREU with *two* rigid equations.

Both problems are highly non-trivial.

8.3 FUTURE WORK

There are several directions for future work and open problems that need to be solved. We can divide these problems into three categories:

1. Classification of unsolved fragments of SREU.
2. Investigation of the f-Skeleton problem.
3. Algorithms for SREU.

The first item is directly related to the corresponding questions about fragments of intuitionistic logic with equality. We now address each of the items and discuss some possible ways to approach them.

Unsolved Fragments

One concern of the thesis is to classify fragments of SREU into either decidable or undecidable ones. In that respect there are two unsolved cases:

1. Decidability of monadic SREU [73]. This is equivalent to the decidability of the prenex fragment of intuitionistic logic with equality restricted to function symbols of arity ≤ 1.
2. Decidability of SREU with *two* rigid equations.

If we adopt a more precise notion of classification (that is standard in the context of classical logic [13]), we can note that the following case is unsolved.

3. Decidability of the $\exists^*$-fragment of intuitionistic logic with equality *without constants*.

It should be emphasized that the question of the decidability of monadic SREU is equivalent to the question of the decidability of SREU with just two unary function symbols or SREU_2 [32]. It is shown in Degtyarev, Matiyasevich and Voronkov [31, 32] that the famous *word equation problem*, also known as *unification under associativity* has a simple reduction to SREU_2. The word equation problem is a hard combinatorial problem that was proven decidable by Makanin in 1977 [96]. There are no known interesting upper bounds of the computational complexity of the word equation problem; the complexity of Makanin's algorithm has several exponents [87]. It is known only that the problem is NP-hard [8]. Hence, if SREU_2 is decidable then there is probably no simple proof of that. One approach to prove the decidability is to try to generalize the decidability proof of the case with *one* unary function symbol [31, 32].

The decidability of SREU with two rigid equations is less important, but intriguing. One might try to prove its undecidability by using some ideas from Schubert [130].

There is of course always the question, whether there are some other syntactical criteria, such as the united one variable property, that guarantee decidability. Answers to such questions might be found by studying the relationships between SREU and automata theoretic extensions of tree automata, such as *tree pushdown automata*, and the decision problems of the latter [20, 62, 126, 129]. We believe that the study of such relationships is important also in the context of developing decision algorithms for fragments of SREU.

The f-Skeleton Poblem

The undecidability of the m-Skeleton problem that arises in Step II of the principal procedure of rigid variable methods, indicates the inadequacy of the formulation of the problem. More "intelligent" strategies are needed for choosing multiplicity and increasing it at Step II. In Voronkov [157], the results of Chapter 4 are used to show that such strategies cannot in general be formula-independent, and the following open problem is posed:

> Does there exist an increasing strategy for multiplicity, for which Step II is decidable?

Algorithms for SREU

The topic that has not been treated in the thesis is the study of semi-decision procedures for SREU, or decision procedures for fragments of SREU, e.g., in the context of theorem proving. In general of course, SREU is undecidable and thus does not have a decision procedure.

In Classical Logic The original refutation procedure for classical logic with equality using SREU, given in Gallier et al [57], is based on the assumption that solutions to a system of rigid equations can be found by combining minimal solutions for the individual rigid equations, which cannot work in general. However, this does not automatically imply incompleteness of the refutation procedure they propose. (It may be the case that if a formula has an m-corroborator then their procedure eventually finds an n-corroborator for some $n > m$, although it fails to find an m-corroborator.) There are a number of publications on the use of SREU in automated reasoning, e.g., the papers [4, 5, 6, 7, 52, 53, 56, 58, 66, 115], some of the results are based on the conjecture that SREU is decidable.

Degtyarev and Voronkov [41, 39] present a calculus $\mathcal{BPE}$ for solving *non-simultaneous* rigid E-unification that is based on "rigid" basic superposition and is an adaption of basic superposition of Bachmair [3], Nieuwenhuis and Rubio [106], to "rigid" variables. Their formalization of rigid basic superposition is close to the one in Nieuwenhuis and Rubio [107]. The calculus $\mathcal{BPE}$ is incomplete for solving rigid E-unification in general, but can be used in tableau-based methods to get a complete calculus for classical logic with equality [41, 39]. (See Schumann [131] for a survey of implementations of tableau-based theorem provers.) It should be investigated if a calculus similar to $\mathcal{BPE}$ can be designed for SREU.

In Intuitionistic Logic In intuitionistic logic with equality, SREU is unavoidable [155, 156]; we address this fact briefly in Chapter 7. This explains why there have been so few attempts to handle equality in theorem proving

in intuitionistic logic. Tammet [142] has implemented a resolution based theorem prover for intuitionistic logic and has plans to include equality there. A non-standard formalization of equality is used in Sahlin, Franzén and Haridi [125]. It is noted in Degtyarev and Voronkov [40] that the same situation arises in other non-classical logics with equality, such as certain modal logics with equality. It is clear that, in order to handle equality in intuitionistic logic and other non-classical logics with a semantics based on Kripke semantics, it is necessary to handle SREU. Currently there are no reasonable semi-decision procedures for SREU, except for ones based on straightforward enumeration [115].

Other Applications Due to the simple reduction from second-order unification to SREU [33, 38], a reasonable semi-decision procedure for SREU might also give new insights into how to deal with the former problem. If monadic SREU is decidable then some algorithm for it may shed some light on the complexity of the word equation problem.

Bibliography

1. P.B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, 1981.

2. M. Baaz. Note on the existence of most general semi-unifiers. In *Arithmetic, Proof Theory and Computation Complexity*, volume 23 of *Oxford Logic Guides*, pages 20–29. Oxford University Press, 1993.

3. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. In D. Kapur, editor, *11th International Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 462–476, Saratoga Springs, NY, USA, June 1992. Springer Verlag.

4. Peter Baumgartner. An ordered theory resolution calculus. In A. Voronkov, editor, *Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of *Lecture Notes in Computer Science*, pages 119–130, 1992.

5. G. Becher and U. Petermann. Rigid unification by completion and rigid paramodulation. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence. 18th German Annual Conference on Artificial Intelligence*, volume 861 of *Lecture Notes in Artificial Intelligence*, pages 319–330, Saarbrücken, Germany, September 1994. Springer Verlag.

6. B. Beckert. A completion-based method for mixed universal and rigid E-unification. In A. Bundy, editor, *Automated Deduction — CADE-12. 12th International Conference on Automated Deduction.*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 678–692, Nancy, France, June/July 1994.

7. B. Beckert and R. Hähnle. An improved method for adding equality to free variable semantic tableaux. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 678–692, Saratoga Springs, NY, USA, June 1992. Springer Verlag.

8. D. Benanav, D. Kapur, and P. Narendran. Complexity of matching problems. *Journal of Symbolic Computations*, 3:203–216, 1987.

9. E.W. Beth. *The Foundations of Mathematics*. North Holland, 1959.

10. W. Bibel. On matrices with connections. *Journal of the Association for Computing Machinery*, 28(4):633–645, 1981.

11. W. Bibel and E. Eder. Methods and calculi for deduction. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, chapter 3, pages 67–182. Oxford University Press, 1993.

12. G. Birkhoff. On the structure of abstract algebras. *Proc. Cambridge Phil. Soc.*, 31:433–454, 1935.

13. E. Börger, E. Grädel, and Yu. Gurevich. *The Classical Decision Problem*. Springer Verlag, 1997.

14. W.S. Brainerd. Tree generating regular systems. *Information and Control*, 14:217–231, 1969.

15. J.R. Büchi and J.B. Wright. Mathematical theory of automata. course notes. Communications Sciences 403, University of Michigan, 1960.

16. A. Chandra, H. Lewis, and J. Makowsky. Embedded implicational dependencies and their inference problem. In *Proc. of 13th Annual ACM Symposium on Theory of Computing (STOC)*, pages 342–354, 1981.

17. A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, January 1981.

18. C.C. Chang and H.J. Keisler. *Model Theory*. North-Holland, Amsterdam, third edition, 1990.

19. Jin-Young Choi. *The Decidability Problem for Rigid E-Unification: A New Proof and Extensions*. PhD thesis, University of Pennsylvania, 1993.

20. J.L. Coquidé, M. Dauchet, R. Gilleron, and S. Vágvölgyi. Bottom-up tree pushdown automata: classification and connection with rewrite systems. *Theoretical Computer Science*, 127:69–98, 1994.

21. B. Courcelle. On recognizable sets and tree automata. In M. Nivat and H. Ait-Kaci, editors, *Resolution of Equations in Algebraic Structures*. Academic Press, 1989.

22. M. Dauchet. Rewriting and tree automata. In H. Comon and J.P. Jouannaud, editors, *Term Rewriting (French Spring School of Theoretical Computer Science)*, volume 909 of *Lecture Notes in Computer Science*, pages 95–113. Springer Verlag, Font Romeux, France, 1993.

23. M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of ground term rewriting systems. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 353–360. IEEE Computer Society Press, 1987.

24. M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems. *Information and Computation*, 88:187–201, 1990.

25. M. Dauchet and S. Tison. Tree automata and decidability in ground term rewriting systems. In *Proc. FCT'85*, volume 199 of *Lecture Notes in Computer Science*, pages 80–84, 1985.

26. M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 242–248. IEEE Computer Society Press, 1990.

27. E. De Kogel. Rigid E-unification simplified. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 918 in Lecture Notes in Artificial Intelligence, pages 17–30, Schloß Rheinfels, St. Goar, Germany, May 1995.

28. Eric De Kogel. *Equational Proofs in Tableaux and Logic Programming*. PhD thesis, Tilburg University, the Netherlands, 1995.

29. A. Degtyarev, Yu. Gurevich, P. Narendran, M. Veanes, and A. Voronkov. The decidability of simultaneous rigid E-unification with one variable. UPMAIL Technical Report 139, Uppsala University, Computing Science Department, March 1997.

30. A. Degtyarev, Yu. Gurevich, and A. Voronkov. Herbrand's theorem and equational reasoning: Problems and solutions. UPMAIL Technical Report 128, Uppsala University, Computing Science Department, September 1996. Appears in the Bulletin of the European Association for Theoretical Computer Science (Vol 60, October 1996).

31. A. Degtyarev, Yu. Matiyasevich, and A. Voronkov. Simultaneous rigid E-unification is not so simple. UPMAIL Technical Report 104, Uppsala University, Computing Science Department, April 1995.

32. A. Degtyarev, Yu. Matiyasevich, and A. Voronkov. Simultaneous rigid E-unification and related algorithmic problems. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 494–502, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

33. A. Degtyarev and A. Voronkov. Reduction of second-order unification to simultaneous rigid E-unification. UPMAIL Technical Report 109, Uppsala University, Computing Science Department, June 1995.

34. A. Degtyarev and A. Voronkov. Simultaneous rigid E-unification is undecidable. UPMAIL Technical Report 105, Uppsala University, Computing Science Department, May 1995.

35. A. Degtyarev and A. Voronkov. Decidability problems for the prenex fragment of intuitionistic logic. In *Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 503–512, New Brunswick, NJ, July 1996. IEEE Computer Society Press.

36. A. Degtyarev and A. Voronkov. Equality elimination for the tableau method. In J. Calmet and C. Limongelli, editors, *Design and Implementation of Symbolic Computation Systems (DISCO'96)*, volume 1128 of *Lecture Notes in Computer Science*, Karlsruhe, Germany, 1996.

37. A. Degtyarev and A. Voronkov. Simultaneous rigid E-unification is undecidable. In H. Kleine Büning, editor, *Computer Science Logic. 9th International Workshop, CSL'95*, volume 1092 of *Lecture Notes in Computer Science*, pages 178–190, Paderborn, Germany, September 1995, 1996.

38. A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1–2):291–300, 1996.

39. A. Degtyarev and A. Voronkov. What you always wanted to know about rigid E-unification. In J.J. Alferes, L. Moniz Pereira, and E. Orlowska, editors, *Logics in Artificial Intelligence (JELIA'96)*, volume 1126 of *Lecture Notes in Artificial Intelligence*, Evora, Portugal, 1996.

40. A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi: a tutorial. UPMAIL Technical Report, Uppsala University, Computing Science Department, 1997. To appear.

41. A. Degtyarev and A. Voronkov. What you always wanted to know about rigid E-unification. UPMAIL Technical Report 143, Uppsala University, Computing Science Department, April 1997.

42. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, chapter 6, pages 243–309. North Holland, Amsterdam, 1990.

43. J. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.

44. C. Dwork, P.C. Kanellakis, and J.C. Mitchell. On the sequential nature of unification. *Journal of Logic Programming*, 1:35–50, 1984.

45. E. Eder. A comparison of the resolution calculus and the connection method, and a new calculus generalizing both methods. In E. Börger, G. Jäger, H. Kleine Büning, and M.M. Richter, editors, *CSL'88 (Proc. 2nd Workshop on Computer Science Logic)*, volume 385 of *Lecture Notes in Computer Science*, pages 80–98. Springer Verlag, 1988.

46. E. Eder. Consolution and its relation with resolution. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 132–136, 1991.

47. M. Fitting. *First Order Logic and Automated Theorem Proving.* Springer Verlag, New York, second edition, 1996.

48. T. Frühwirth, E. Shapiro, M. Vardi, and E. Yardeni. Logic programs as types of logic programs. In *Proc. 6th Symposium on Logics in Computer Science (LICS)*, pages 300–309, 1991.

49. M. Fürer. The complexity of the inequivalence problem for regular expressions with intersection. In *Proc. 7th International Colloquium on Autamata, Languages, and Programming (ICALP)*, volume 85 of *Lecture Notes in Computer Science*, pages 234–245, New York, 1980. Springer Verlag.

50. J. Gallier. Unification procedures in automated deduction methods based on matings: a survey. In M. Nivat and A. Podelski, editors, *Tree Automata and Languages*, pages 439–485. Elsevier Science, 1992.

51. J. Gallier, P. Narendran, D. Plaisted, S. Raatz, and W. Snyder. An algorithm for finding canonical sets of ground rewrite rules in polynomial time. *Journal of the Association for Computing Machinery*, 40(1):1–16, 1993.

52. J. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification: NP-completeness and applications to equational matings. *Information and Computation*, 87(1/2):129–195, 1990.

53. J. Gallier, P. Narendran, S. Raatz, and W. Snyder. Theorem proving using equational matings and rigid E-unification. *Journal of the Association for Computing Machinery*, 39(2):377–429, 1992.

54. J.H. Gallier and R.V. Book. Reductions in tree replacement systems. *Theoretical Computer Science*, 37:123–150, 1985.

55. J.H. Gallier, P. Narendran, D. Plaisted, S. Raatz, and W. Snyder. Finding canonical rewrite systems equivalent to a finite set of ground equations in polynomial time. In *CADE'88 (9th Int. Conf. on Automated Deduction)*, Lecture Notes in Computer Science, Argonne, Illinois, May 1988.

56. J.H. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification is NP-complete. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 338–346. IEEE Computer Society Press, July 1988.

57. J.H. Gallier, S. Raatz, and W. Snyder. Theorem proving using rigid E-unification: Equational matings. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 338–346. IEEE Computer Society Press, 1987.

58. J.H. Gallier, S. Raatz, and W. Snyder. Rigid E-unification and its applications to equational matings. In H. Aït Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 1, pages 151–216. Academic Press, 1989.

59. H. Galperin and A. Wigderson. Succinct representations of graphs. *Information and Control*, 56:183–198, 1983.

60. F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiodó, Budapest, 1984.

61. G. Gentzen. Investigation into logical deduction. In M.E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland, 1969.

62. R. Gilleron. Decision problems for term rewriting systems and recognizable tree languages. In *8th Annual Symposium on Theoretical Aspects of Computer Science (STACS'91)*, volume 480 of *Lecture Notes in Computer Science*, pages 148–159, 1991.

63. K. Gödel. *Collected Works, Vol I: Publications 1929–1936*, pages 144–195. Oxford University Press, 1986.

64. W.D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

65. J. Goubault. Simultaneous rigid E-unification is NEXPTIME-complete (correction). Technical Report 93047, Bull, 1993.

66. J. Goubault. Rigid $\vec{E}$-unifiability is DEXPTIME-complete. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*. IEEE Computer Society Press, 1994.

67. R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. A compendium of problems complete for P. Technical Report TR 91–05–01, Department of Computer Science and Engineering, University of Washington, 1991.

68. R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995.

69. Y. Gurevich. Formulas with one $\forall$. In *Selected Questions in Algebra and Logic; in memory of A. Malćev*, pages 97–110. Nauka, Moscow, 1973. In Russian. A German translation is available at TIB Universität Hannover, Germany.

70. Y. Gurevich. The decision problem for standard classes. *Journal of Symbolic Logic*, 41(2):460–464, 1976.

71. Y. Gurevich. On the classical decision problem. In G. Rozenberg and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, pages 254–265. World Scientific, 1993. Originally published in the Bull. of European Association for Theoretical Computer Science, Oct. 1990, 140–150.

72. Y. Gurevich and M. Veanes. Some undecidable problems related to the Herbrand theorem. UPMAIL Technical Report 138, Uppsala University, Computing Science Department, March 1997.

73. Y. Gurevich and A. Voronkov. The monadic case of simultaneous rigid E-unification. UPMAIL Technical Report 137, Uppsala University, Computing Science Department, 1997. To appear in Proc. of *ICALP'97*.

74. J. Herbrand. *Logical Writings*. Harvard University Press, 1972.

75. J.R. Hindley and J.P. Seldin. *Introduction to Combinatorics and λ-Calculus*. Cambridge University Press, 1986.

76. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Co., 1979.

77. J. Hudelmaier. An $O(n \log n)$-space decision procedure for intuitionistic propositional logic. *Journal of Logic and Computation*, 3(1):63–75, 1993.

78. G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the Association for Computing Machinery*, 27(4):797–821, October 1980.

79. H.B. Hunt III. The equivalence problem for regular expressions with intersection is not polynomial in tape. Technical Report TR 73-161, Cornell University, Ithaca, NY., 1973.

80. N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.

81. D.S. Johnson. A catalog of complexity classes. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity, chapter 2, pages 67–161. Elsevier Science, Amsterdam, 1990.

82. N.D. Jones and W.T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3(2):105–117, 1976.

83. S. Kanger. *Provability in Logic*, volume 1 of *Studies in Philosophy*. Almquist and Wicksell, Stockholm, 1957.

84. S. Kanger. A simplified proof method for elementary logic. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 364–371. Springer Verlag, 1983. Originally appeared in 1963.

85. A.J. Kfoury, J. Tiuryn, and P. Urzyczyn. The undecidability of the semi-unification problem. *Information and Computation*, 102:83–101, 1993.

86. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.

87. A. Kościelski and L. Pacholski. Complexity of Makanin's algorithm. *Journal of the Association for Computing Machinery*, 43(4):670–684, 1996.

88. D. Kozen. Complexity of finitely presented algebras. Technical Report TR 76-294, Cornell University, Ithaca, N.Y., 1976.

89. D. Kozen. Complexity of finitely presented algebras. In *Proc. of the 9th Annual Symposium on Theory of Computing*, pages 164–177, New York, 1977. ACM.

90. D. Kozen. Lower bounds for natural proof systems. In *Proc. 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 254–266, 1977.

91. D. Kozen. Positive first-order logic is NP-complete. *IBM J. of Research and Development*, 25(4):327–332, 1981.

92. D.S. Lankford. Canonical inference. Technical report, Department of Mathematics, South-Western University, Georgetown, Texas, 1975.

93. V. Lifschitz. Problem of decidability for some constructive theories of equalities (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 4:78–85, 1967. English Translation in: Seminars in Mathematics: Steklov Math. Inst. 4, Consultants Bureau, NY-London, 1969, p.29–31.

94. A. Lozano and J.L. Balcázar. The complexity of graph problems for succinctly represented graphs. In M. Nagl, editor, *Graph-Theoretic Concepts in Computer Science, 15th International Workshop WG'89*, volume 411 of *Lecture Notes in Computer Science*, pages 277–286. Springer Verlag, 1989.

95. M. Magidor and G. Moran. Finite automata over finite trees. Technical Report 30, Hebrew University, Jerusalem, 1969.

96. G.S. Makanin. The problem of solvability of equations in free semigroups. *Mat. Sbornik (in Russian)*, 103(2):147–236, 1977. English Translation in American Mathematical Soc. Translations (2), vol. 117, 1981.

97. A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, 1982.

98. S. Maslov and V. Orevkov. Decidable classes reducing to one-quantifier class. In *Proc. Steklov Inst. Steklov Math*, volume 121, pages 61–72, 1972. Russian original in Trudy Math. Inst. Steklov.

99. Yu.V. Matiyasevič. The diophantiness of recursively enumerable sets (in Russian). *Soviet Mathematical Doklady*, pages 279–282, 1970.

100. A.J. Mayer and L.J. Stockmeyer. The complexity of word problems – this time with interleaving. *Information and Computation*, 115:293–311, 1994.

101. A.R. Meyer and M.J. Fisher. Economy of description of automata, grammars and formal systems. In *Proc. 12th IEEE Symposium on Switching and Automata Theory (SWAT)*, pages 188–191, 1971.

102. A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. 13th IEEE Symposium on Switching and Automata Theory (SWAT)*, pages 125–129, 1972.

103. J. Mezei and J.B. Wright. Algebraic automata and context-free sets. *Information and Control*, 11:3–29, 1967.

104. G.E. Mints. Choice of terms in quantifier rules of constructive predicate calculus (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 4:78–85, 1967. English Translation in: Seminars in Mathematics: Steklov Math. Inst. 4, Consultants Bureau, NY-London, 1969, p.43–46.

105. E.F. Moore. Gedanken experiments on sequential machines. In *Automata Studies*, pages 129–153. Princeton Univ. Press, Princeton, N.J., 1956.

106. R. Nieuwenhuis and A. Rubio. Basic superposition is complete. In *ESOP'92*, volume 582 of *Lecture Notes in Computer Science*, pages 371–389. Springer Verlag, 1992.

107. R. Nieuwenhuis and A. Rubio. Theorem proving with ordering and equality constrained clauses. *Journal of Symbolic Computations*, 19:321–351, 1995.

108. W.M.J. Ophelders and H.C.M. de Swart. Tableaux versus resolution; a comparison. *Fundamenta Informaticae*, 18:109–127, 1993.

109. V.P. Orevkov. Unsolvability in the constructive predicate calculus of the class of the formulas of the type $\neg\neg\forall\exists$ (in Russian). *Soviet Mathematical Doklady*, 163(3):581–583, 1965.

110. V.P. Orevkov. On nonlengthening applications of equality rules (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 16:152–156, 1969. English Translation in: Seminars in Mathematics: Steklov Math. Inst. 16, Consultants Bureau, NY-London, 1971, p.77-79.

111. V.P. Orevkov. Solvable classes of pseudo-prenex formulas (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 60:109–170, 1976. English translation in: Journal of Soviet Mathematics.

112. C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

113. C.H. Papadimitriou and M. Yannakakis. A note on succinct representations of graphs. *Information and Control*, 71:181–185, 1986.

114. M. Paterson and M. Wegman. Linear unification. *Journal of Computer and System Sciences*, 16:158–167, 1978.

115. U. Petermann. A complete connection calculus with rigid E-unification. In *JELIA'94*, volume 838 of *Lecture Notes in Computer Science*, pages 152–166, 1994.

116. D.A. Plaisted. Special cases and substitutes for rigid E-unification. Technical Report MPI-I-95-2-010, Max-Planck-Institut für Informatik, November 1995.

117. E.L. Post. A variant of a recursively unsolvable problem. *Bull. Am. Math. Soc.*, 52(4):264–268, 1946.

118. D. Prawitz. A proof procedure with matrix reduction. In M. Laudet, L. Lacombe, L. Nolin, and M Schutzenberger, editors, *Symp. on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 207–213. Springer Verlag, 1970.

119. D. Prawitz. An improved proof procedure. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 162–201. Springer Verlag, 1983. Originally appeared in 1960.

120. W.V. Quine. A proof procedure for quantification theory. *Journal of Symbolic Logic*, 20:141–149, 1955.

121. M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

122. M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

123. G. Robinson and L.T. Wos. Paramodulation and theorem-proving in first order theories with equality. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 135–150. Edinburgh University Press, 1969.

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

125. D. Sahlin, T. Franzén, and S. Haridi. An intuitionistic predicate logic theorem prover. *Journal of Logic and Computation*, 2(5):619–656, 1992.

126. K. Salomaa. Deterministic tree pushdown automata and monadic tree rewriting systems. *Journal of Computer and System Sciences*, 37:367–394, 1988.

127. K. Salomaa, D. Wood, and S. Yu. Complexity of $E0L$ structural equivalence. In *Mathematical Foundations of Computer Science 1994*, number 841 in Lecture Notes in Computer Science, pages 587–596, Košice, Slovakia, 1994. Springer Verlag.

128. W.J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

129. K.M. Schimpf and J.H. Gallier. Tree pushdown automata. *Journal of Computer and System Sciences*, 30:25–40, 1985.

130. A. Schubert. Second-order unification and type inference for Church-style polymorphism. Technical report, Institute of Informatics, Warsaw University, January 1997.

131. J. Schumann. Tableau-based theorem provers: Systems and implementations. *Journal of Automated Reasoning*, 13(3):409–421, 1994.

132. H. Seidl. Deciding equivalence of finite tree automata. *SIAM Journal of Computing*, 19(3):424–437, 1990.

133. H. Seidl. Haskell overloading is DEXPTIME-complete. *Information Processing Letters*, 52(2):57–60, 1994.

134. E.Y. Shapiro. Alternation and the computational complexity of logic programs. *Journal of Logic Programming*, 1:19–33, 1984.

135. S. Shelah. Decidability of a portion of the predicate calculus. *Israel Journal of Math.*, 28:32–44, 1977.

136. R. Shostak. An algorithm for reasoning about equality. *Communications of the ACM*, 21:583–585, July 1978.

137. R.M. Smullyan. *First-Order Logic*. Springer Verlag, 1968. Revised edition, Dover Press, New York, 1994.

138. W. Snyder. Efficient ground completion: An $O(n\log n)$ algorithm for generating reduced sets of ground rewrite rules equivalent to a set of ground equations E. In G. Goos and J. Hartmanis, editors, *Rewriting Techniques and Applications*, volume 355 of *Lecture Notes in Computer Science*, pages 419–433. Springer-Verlag, 1989.

139. R. Statman. Lower bounds on Herbrand's theorem. *Proc. American Mathematical Society*, 75(1):104–107, 1979.

140. R. Stearns and H. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. In *Proc. 22th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 74–81, 1981.

141. R. Stearns and H. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal of Computing*, 14:598–611, 1985.

142. T. Tammet. A resolution theorem prover for intuitionistic logic. In *Proceedings of the Conference on Automated Deduction (CADE-13)*, volume 1104 of *Lecture Notes in Computer Science*, pages 2–16, 1996.

143. J.W. Thatcher and J.B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.

144. Wolfgang Thomas. Languages, automata and logic. Technical Report 9607, Institut für Informatik und Praktische Mathematik der Christian-Albrechts-Universität zu Kiel, 1996. A modified version will appear as a chapter of the *Handbook of Formal Language Theory*, Springer-Verlag, edited by G. Rozenberg and A. Salomaa.

145. A.S. Troelstra and D. van Dalen. *Constructivism in Mathematics An Introduction, Volume I*, volume 121 of *Studies In Logic and the Foundations of Mathematics*. North-Holland, 1988.

146. M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symposium on Theory of Computing (STOC)*, pages 137–146, 1982.

147. M. Veanes. Uniform representation of recursively enumerable sets with simultaneous rigid E-unification. UPMAIL Technical Report 126, Uppsala University, Computing Science Department, July 1996.

148. M. Veanes. Uniform representation of recursively enumerable sets with simultaneous rigid E-unification (extended abstract). In K.U. Schulz and S. Kepser, editors, *Extended Abstracts of the Tenth International Workshop on Unification UNIF'96*, number CIS-Bericht-96-91, pages 7–15. Munich University, 1996.

149. M. Veanes. On computational complexity of basic decision problems of finite tree automata. UPMAIL Technical Report 133, Uppsala University, Computing Science Department, January 1997.

150. M. Veanes. The undecidability of simultaneous rigid E-unification with two variables. To appear in *Proc. Kurt Gödel Colloquium KGC'97*, 1997.

151. P.J. Voda and J. Komara. On Herbrand skeletons. Technical report, Institute of Informatics, Comenius University Bratislava, July 1995. Revised January 1996.

152. A. Voronkov. On proof-search in intuitionistic logic with equality, or back to simultaneous rigid E-Unification. UPMAIL Technical Report 121, Uppsala University, Computing Science Department, January 1996.

153. A. Voronkov. Proof search in intuitionistic logic based on constraint satisfaction. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods. 5th International Workshop, TABLEAUX '96*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 312–329, Terrasini, Palermo Italy, May 1996.

154. A. Voronkov. Proof-search in intuitionistic logic based on the constraint satisfaction. UPMAIL Technical Report 120, Uppsala University, Computing Science Department, January 1996. Updated March 11, 1996.

155. A. Voronkov. Proof search in intuitionistic logic with equality, or back to simultaneous rigid E-unification. In M.A. McRobbie and J.K. Slaney, editors, *Automated Deduction — CADE-13*, volume 1104 of *Lecture Notes in Computer Science*, pages 32–46, New Brunswick, NJ, USA, 1996.

156. A. Voronkov. Proof-search in intuitionistic logic with equality, or back to simultaneous rigid E-unification. *Journal of Automated Reasoning*, 1997. To appear.

157. A. Voronkov. Rigid variables considered harmful. UPMAIL Technical Report 134, Uppsala University, Computing Science Department, January 1997. To appear in Proc. *IJCAI'97*.

158. H. Wang. Towards mechanical mathematics. *IBM J. of Research and Development*, 4:2–22, 1960.

159. M. Wirsing. Das entscheidungsproblem der klasse von formel, die höchstens zwei primformeln enthalten. *Manuscripta Mathematica*, 22:13–25, 1977.

160. M. Wirsing. Kleine unentscheidbare klassen der prädikatenlogic mit identität und funktionszeichen. *Archiv math. Logik u. Grundlagenforschung*, 19:97–109, 1978.

161. H. Yasuura. On parallel computational complexity of unification. In *Proc. International Conference on Fifth Generation Computer Systems*, pages 235–243, 1984.

Index