# Extensions of Presburger Arithmetic and Model Checking One-Counter Automata


Antonia Lechner

Oriel College

University of Oxford

A thesis submitted for the degree of

*Doctor of Philosophy*

Trinity 2016

# Acknowledgements

I would like to thank my supervisor, James Worrell, for all his advice, guidance and encouragement that helped me in writing this thesis.

I am grateful to the Engineering and Physical Sciences Research Council (EPSRC) for generously providing me with a scholarship during the majority of the time of my research. My thanks also go to Jean-François Raskin, who accepted me to his research group and organised the funding I needed in those last few months of finalising my thesis.

I would further like to thank Vojtěch Forejt and Gilles Geeraerts, the examiners for this thesis, for taking the time to read through it in depth and make valuable suggestions for how to improve it. Thank you also to Joël Ouaknine, who contributed to many helpful discussions about various research topics.

I am indebted to my maths teacher, who encouraged me to apply to Oxford for my undergraduate degree – a decision I would never have made otherwise.

To all of my friends and colleagues who have helped me along this journey, by proofreading my work and providing feedback, coming up with ideas for solving new problems, giving advice on research techniques and career options, or even just listening to me complain at times when I seemed to be making little progress and cheering me up when I most needed it: thank you. To my best friend and partner, thank you for always being there for me.

My sincere thanks go to my grandma, who was an example for me in many ways. Finally I would like to thank my parents, who have always supported me throughout my life and academic career. I could not have achieved this without their help.

# Abstract

This thesis concerns decision procedures for fragments of linear arithmetic and their application to model-checking one-counter automata. The first part of this thesis covers the complexity of decision problems for different types of linear arithmetic, namely the existential subset of the first-order linear theory over the $p$-adic numbers and the existential subset of Presburger arithmetic with divisibility, with all integer constants and coefficients represented in binary. The most important result of this part is a new upper complexity bound of **NEXPTIME** for existential Presburger arithmetic with divisibility. The best bound that was known previously was **2NEXPTIME**, which followed directly from the original proof of decidability of this theory by Lipshitz in 1976. Lipshitz also gave a proof of **NP**-hardness of the problem in 1981. Our result is the first improvement of the bound since this original description of a decision procedure.

Another new result, which is both an important building block in establishing our new upper complexity bound for existential Presburger arithmetic with divisibility and an interesting result in its own right, is that the decision problem for the existential linear theory of the $p$-adic numbers is in the Counting Hierarchy **CH**, and thus within **PSPACE**. The precise complexity of this problem was stated as open by Weispfenning in 1988, who showed that it is in **EXPTIME** and **NP**-hard.

The second part of this thesis covers two problems concerning one-counter automata. The first problem is an LTL synthesis problem on one-counter automata with integer-valued and parameterised updates and with equality tests. The decidability of this problem was stated as open by Göller et al. in 2010. We give a reduction of this problem to the decision problem of a subset of Presburger arithmetic with divisibility with one quantifier alternation and a restriction on existentially quantified variables. A proof of decidability of this theory is currently under review.

The final result of this thesis concerns a type of one-counter automata that differs from the previous one in that it allows parameters only on tests, not actions, and it includes both equality and disequality tests on counter values. The decidability of the basic reachability problem on such one-counter automata was stated as an open problem by Demri and Sangnier in 2010. We show that this problem is decidable by a reduction to the decision problem for Presburger arithmetic.

# Contents

# Chapter 1

# Introduction

## 1.1 Background

Counter automata are an important type of infinite-state computing device with numerous applications especially in formal verification. In general, a counter automaton is a finite automaton which operates on a collection of counters, each of which stores a single (usually integer) value. Counter automata are often used as computational models for infinite-state systems, and in general they are equivalent to Turing machines, as shown by Minsky in [Min61]. However, various problems have been shown to be decidable for special classes of counter automata, such as reversal-bounded counter automata [ISD+02], flat counter automata [FL02], vector addition systems with states [Reu90], and one-counter automata [Iba79]. In particular, the reachability problem (given a counter automaton in state $v$ with counter values $\boldsymbol{c}$, is there a computation that ends up in state $v'$ with counter values $\boldsymbol{c}'$) is decidable for all of the above models.

Many applications have been found for counter automata, ranging from modelling resource-bounded systems and programs with lists [BBH+06] to XML query evaluation [CR04] and broadcast protocols [EFM99]. While in their original description by Minsky, the only actions on counters were defined to be simple incrementation and decrementation operations, more general kinds of updates have since been considered due to their wide range of modern applications, including addition of integers and parameters as well as counter resets and transfers.

Counter automata can be considered as language acceptors with a read-only input tape [ID04]. In a similar vein one can consider counter automata with parameters (or read-only input registers) and ask for which parameter values a given automaton has an accepting computation [IJTW93].

In this thesis, we will be particularly interested in nondeterministic one-counter automata, which operate over only a single counter. Various classes of one-counter automata have been studied in the literature; a classical version of nondeterministic one-counter automata has a counter that ranges over the nonnegative integers and includes integer-valued additions to the counter value as well as transitions that test if the current counter value is equal to zero. We will consider several generalisations of such classical one-counter automata in this thesis, with extensions like parameterised counter updates or parameterised equality or disequality tests. We assume that all integers in the description of one-counter automata are represented in binary.

Various links have been discovered between verification and synthesis problems for counter automata and first-order linear theories over the integers. A first-order linear theory, or linear arithmetic, is a first-order theory which includes addition of constants and variables and multiplication by constants, but no multiplication of two or more variables. A classical example is Presburger arithmetic [Pre29], which is defined over the integers and includes an addition operation and an order relation, so that all atomic formulas of this theory are equalities or inequalities between linear polynomials with integer coefficients.

The decision problem for Presburger arithmetic was shown to be decidable by Presburger [Pre29], with Fischer and Rabin later proving a doubly exponential lower complexity bound [FR74]. The purely existential fragment of Presburger arithmetic is in **NP**, as can be seen for example from a result in [vzGS78]. When Presburger arithmetic is augmented with a general multiplication operation (i.e., multiplication by variables and not just constants), the resulting language is undecidable even when restricted to its purely existential fragment. This follows immediately from the undecidability of Hilbert's tenth problem, famously proved by Matiyasevich [Mat70]. The language obtained by adding a binary divisibility predicate $\mid$ to Presburger arithmetic, where $a \mid b$ if and only if $ac = b$ for some integer $c$ (we will call this language Presburger arithmetic with divisibility), is also undecidable, since multiplication can be encoded in this language even when it is restricted to only one quantifier alternation, as shown by Robinson [Rob49].

In 1976, Lipshitz [Lip76] and Bel'tyukov [Bel76] independently showed that the truth of *existential* sentences of Presburger arithmetic with divisibility is decidable. This result can be seen as a generalisation of the Chinese remainder theorem, a fundamental theorem in number theory, so it is not surprising that this decidability result has since found numerous applications, particularly in automata theory: it has

been used to establish the decidability of verification problems for one-counter automata [GHOW10, GI82, HKOW09, ID04, ID06], parametric timed automata [BO14], and semilinear automata [BL12]. Moreover, in theorem proving it has been used to prove decidability of a subcase of the simultaneous rigid $E$-unification problem for first-order logic [DMV96].

Despite the large number of applications of the decidability result for existential Presburger arithmetic with divisibility, the precise complexity of the decision problem has been an open problem since 1976. It follows from the algorithm described in [Lip76] that the problem is in **2NEXPTIME**, and Lipshitz further showed in 1981 that the problem is **NP**-hard even in the restricted case of formulas which consist of only one conjunction of five divisibility atoms [Lip81].

This large complexity gap is at odds with a folklore belief that existential Presburger arithmetic with divisibility is **NP**-complete; see, for instance, [BO14, Sec. 2.1] and [Haa12, Thm. 2.6.3], both of which cite [Lip81] as source. One reason for this misunderstanding of the upper complexity bound may well be the considerable mathematical depth and intricacy of Lipshitz's proof, making it difficult to read and understand. Another cause may perhaps be traced to a typo in Chapter 10 of the Handbook of Automated Reasoning, which contains the sentence "[...] the Diophantine problem for addition and divisibility [...] whose complexity is now known [Lipshitz 1981]" [DV01, p. 651]. The authors had clearly intended to write "not known" instead, as witnessed, for example, by another earlier paper of theirs in which they state that "it is not known whether the Diophantine problem for addition and divisibility is in **NP** [Lipshitz 81]" [DMV95, p. 23].

An interesting observation is that the same complexity bounds of **NP** and **2NEXPTIME** were shown for the following two problems, by nondeterministic polynomial-time reductions in both directions with the decision problem for existential Presburger arithmetic with divisibility:

- the reachability problem in parameterised one-counter automata with integer-valued and parameterised counter updates and equality tests [HKOW09][1], and
- the subcase of the simultaneous rigid $E$-unification problem for first-order logic which includes only one unary function symbol (and any number of constants) [DMV96].

[1]The reduction shown in [HKOW09] can be modified slightly to be in nondeterministic polynomial time. We will describe details of this in Chapter 5.

The nondeterministic polynomial-time reductions show that any improvement of the complexity bounds for one of these three problems would immediately give the same result for the remaining two.

The connection of Presburger arithmetic and its extensions to one-counter automata is the main motivating factor behind the research presented in this thesis. (The connections to the simultaneous rigid $E$-unification problem lie outside the scope of this thesis.)

Moreover, we also consider the connection between existential Presburger arithmetic with divisibility and the existential subset of the first-order linear theory of the $p$-adic numbers. The system of $p$-adic numbers was first described by Hensel in [Hen97]. The $p$ in $p$-adic numbers stands for a fixed but arbitrary prime number. The $p$-adic numbers are a completion of the rational numbers which is analogous to the real numbers, but using a different metric, where two numbers are interpreted to be "close" to each other if their difference is divisible by a high power of $p$.

In his original proof of decidability for existential Presburger arithmetic with divisibility, Lipshitz reduced the general problem to that of deciding formulas in a normal form (involving only divisibilities and no Presburger conjuncts), which he in turn reduced to the decision problem for the existential linear theory of the $p$-adic numbers for a given prime $p$. This theory has two types of variables: integer-valued and $p$-adic variables. It includes addition on both integer-valued and $p$-adic terms, an order on integer-valued terms, and a unary valuation function from $p$-adic terms to integer-valued terms. This theory was later studied in detail by Weispfenning in [Wei88], where he proved an upper complexity bound of **EXPTIME** and a lower bound of **NP** for the associated decision problem, stating the precise complexity of this problem as open.

## 1.2 Results and Thesis Structure

In this thesis, we present new results concerning both linear arithmetic and one-counter automata, emphasising the connection between these topics.

We will start by studying the existential linear theory of the $p$-adic numbers. To improve the **EXPTIME** upper complexity bound shown by Weispfenning, we first apply some simple transformations to formulas so that the valuation function appears only in equalities, and not inequalities. We then use a quantifier elimination procedure to eliminate all variables of $p$-adic sort in turn, with the result being a formula in Presburger arithmetic.

By representing rational numbers in these computations as pairs of arithmetic circuits, we can use recent complexity results about problems related to arithmetic circuits to show that the decision problem for the existential linear theory of the $p$-adic numbers is in **CH**, i.e., the Counting Hierarchy, which is contained in **PSPACE**. To the best of our knowledge, this new upper complexity bound is the first major advance on this problem since the publication of Weispfenning's original paper. The problem is now known to be between **NP** and **CH**, with the precise complexity still being an open problem.

These new results then find immediate use as a building block in the proof of our second main result, an improved upper complexity bound of **NEXPTIME** for the decision problem for existential Presburger arithmetic with divisibility. We achieve this bound by carefully analysing the original proof of decidability by Lipshitz, identifying a cause of an exponential blow-up in this decision procedure, and adapting the proof to avoid this blow-up. This way we show that every quantifier-free formula of Presburger arithmetic with divisibility has a solution of size at most singly exponential in the size of the formula. We also give an example of a formula whose smallest solution is of singly exponential size, showing that our new upper bound on the solution size is optimal. As a corollary, we immediately derive our **NEXPTIME** upper bound on the complexity of the corresponding decision problem, improving Lipshitz's complexity result for deciding the truth of formulas of existential Presburger arithmetic with divisibility by a full exponential.

We can conclude from this new result that both the reachability problem for parameterised one-counter automata with equality tests [HKOW09] and the case of simultaneous rigid $E$-unification where the signature contains only one unary function symbol [DMV95] are also in **NEXPTIME**.

In the second part of this thesis, we work with two different types of one-counter automata. The first type is a (fully) parameterised version of classical one-counter automata, which allows adding integers or parameters to the counter and testing if the counter value is zero. It is easy to see that integer or parameter valued equality tests can be encoded in this model. The reasons why we only explicitly include zero tests in the definition are that it follows classical definitions of counter automata and that it simplifies some of the resulting proofs.

The second type of one-counter automata has both integer-valued and parameterised equality and disequality tests, but does not allow parameter-valued counter updates. Thus it is more general than the first type in the tests on counter values that it allows, but more specific in its counter updates. To distinguish them from

the fully parameterised (in that both actions and guards may include parameters) first type, we will call these *one-counter automata with parameterised tests*, rather than *parameterised one-counter automata*. The two models do not appear to be easily interreducible.

First we will consider parameterised one-counter automata with equality tests. [HKOW09] showed that the reachability problem for this model can be reduced to the decision problem for the existential fragment of Presburger arithmetic with divisibility. Some small modifications to their proof can be made to obtain a nondeterministic polynomial-time complexity of the reduction. The problem that we are interested in for this model is an LTL (linear temporal logic) synthesis problem. Given a parameterised one-counter automaton, an initial configuration of the automaton, and an LTL formula, we seek to determine if there exist values which, when substituted for the parameters, yield an automaton in which all computations starting in the initial configuration satisfy the LTL formula. This problem is motivated by [GHOW10], where it was shown that the corresponding problem for CTL is undecidable, and the LTL case was stated as an open problem. We use a modification of the translation from [HKOW09] of reachability to the decision problem for existential Presburger arithmetic with divisibility to reduce the LTL synthesis problem for parameterised one-counter automata to the decision problem for a subset of Presburger arithmetic with divisibility that allows one quantifier alternation and has certain restrictions on divisibilities. The decision problem for this logic was stated to be decidable in [BI05]. A gap in their proof was recently found and the decidability result is currently under review.

Our final result is a decidability proof of model checking a temporal logic called flat Freeze LTL on one-counter automata. This logic extends LTL with a freeze operator which, when interpreted over computations of one-counter automata, can store the current counter value in registers and compare registers with a later counter value. The model checking problem for the full set of Freeze LTL formulas was shown to be undecidable even for classical one-counter automata including only integer updates and zero tests (and no parameters or disequality tests) [DLS08].

In [DS10], the authors considered the decidability of model checking the flat fragment of Freeze LTL across a range of different counter automata models. The flatness requirement adds certain restrictions as to where freeze operators may occur in formulas. The approach taken in [DS10] was to reduce the model checking problem for flat Freeze LTL on a class of counter automata to a repeated reachability problem in counter automata of the same class which also include parameterised equality

and disequality tests. This problem, called the *generalised repeated control-state reachability* problem, asks whether there exist values for the parameters of such an automaton such that several sets of accepting states are visited infinitely many times.

This repeated reachability problem was shown to be decidable for the class of reversal-bounded counter automata with parameterised equality and disequality tests in [DS10], implying decidability of the model checking problem for flat Freeze LTL on reversal-bounded counter automata. The decidability of the repeated reachability problem for one-counter automata with parameterised equality and inequality tests (which would show the decidability of model checking flat Freeze LTL on classical one-counter automata) was then stated as an open problem in the same paper. In fact, even the decidability of the basic reachability problem (is there a computation from a given initial configuration to a given target configuration) has been open for one-counter automata with these kinds of parameterised tests. We show that the basic reachability problem as well as the repeated reachability problem are decidable, by reducing both problems to the decision problem for Presburger arithmetic, and obtain as a corollary the decidability of model checking flat Freeze LTL on classical, non-parameterised one-counter automata with equality tests.

In summary, the main results of this thesis are:

- that the decision problem for the existential linear theory of the $p$-adic numbers lies in the Counting Hierarchy **CH**, and thus in **PSPACE**. (Chapter 3)
- that every satisfiable quantifier-free formula $\varphi$ in Presburger arithmetic with divisibility has a solution of size at most $|\varphi|^{n^{O(1)}}$, where $n$ is the number of variables and $|\varphi|$ is the size of the formula. It follows that deciding formulas of existential Presburger arithmetic with divisibility is in **NEXPTIME**. (Chapter 4)
- a reduction of the LTL synthesis problem on parameterised one-counter automata with equality tests to a subset of Presburger arithmetic with divisibility claimed to be decidable in [BI05]. (Chapter 5)
- decision procedures for basic reachability and repeated reachability in one-counter automata with parameterised equality and disequality tests, implying decidability of the model checking problem for flat Freeze LTL on classical one-counter automata. (Chapter 6)

Chapter 2 introduces some global preliminaries, which will be required in more than one of the following chapters. In addition, each of the following chapters also starts with a short section presenting preliminaries local to that chapter.

The results from Chapters 3 and 4 were published at LICS 2015, in a paper co-authored by Joël Ouaknine and James Worrell [LOW15]. Chapter 5 is based on a paper published at RP 2015 [Lec15]. The results from Chapter 6 were published at CONCUR 2016, with co-authors Richard Mayr, Joël Ouaknine, Amaury Pouly and James Worrell [LMO+16].

# Chapter 2

# Preliminaries

## 2.1 Notation

We write $\mathbb{Z}$ for the integers, $\mathbb{N}$ for the natural numbers (i.e., nonnegative integers), $\mathbb{Q}$ for the rational numbers, and $\mathbb{R}$ for the real numbers.

For any integer or rational number, vector, matrix, formula, or automaton $X$, we write $|X|$ for the *size* of $X$, that is, the length of its representation. Unless otherwise specified, we generally assume that integers are represented in binary.

We denote by $(a, b)$ the greatest common divisor of two integers $a$ and $b$, and we write $a \mid b$ to express that $a$ divides $b$, that is, $b$ is an integer multiple of $a$. Note that by this definition of divisibility, $a \mid 0$ for all $a \in \mathbb{Z}$ and $0 \nmid a$ for all $a \in \mathbb{Z} \setminus \{0\}$.

## 2.2 Linear Arithmetic

A *linear arithmetic* is a first-order theory whose terms are linear polynomials. A classical example is Presburger arithmetic. The following definitions form the basis for Chapters 3 and 4, where we will study two different extensions of Presburger arithmetic in detail.

### 2.2.1 Presburger Arithmetic

*Presburger arithmetic* (PA) is the first-order theory over the structure $\langle \mathbb{N}, +, 0, 1 \rangle$, where $\mathbb{N}$ is the set of natural numbers, $+$ is the addition function on natural numbers, and the constants 0 and 1 have their standard meaning. PA was introduced and its decision problem shown to be decidable by Mojżesz Presburger in 1929 [Pre29].

Atomic formulas are equalities between linear polynomials with integer coefficients. For example,

$$3x_1 - 4 = -2x_2$$

is equivalent to

$$3x_1 + 2x_2 = 4,$$

which we use as a shorthand for the atomic PA formula

$$x_1 + x_1 + x_1 + x_2 + x_2 = 1 + 1 + 1 + 1.$$

Similarly, we can express inequalities between linear polynomials using existential quantifiers:

$$2x_1 < x_2$$

can be expressed in PA as

$$\exists y\,(x_1 + x_1 + y + 1 = x_2).$$

Note that the correctness of the last formula depends on $y$ being a nonnegative variable. If variables were to range over all integers, inequalities could not be encoded. However, it is easy to see that if the ordering relation $\leq$ is given as part of the signature, then every formula including nonnegative integer variables is equivalent to one with integer variables and vice versa. Formally, the decision problems for the first-order theory over $\langle \mathbb{N}, +, \leq, 0, 1\rangle$ and that over $\langle \mathbb{Z}, +, \leq, 0, 1\rangle$ are logspace interreducible, meaning that each structure can be interpreted in the other. This is because on the one hand, we can always add a subformula of the form $x_i \geq 0$ to make a variable nonnegative, and on the other hand, every integer variable can be expressed as the difference of two nonnegative integer variables.

From now on we will assume, using shorthand notation, that the atomic formulas of PA are equalities and inequalities between linear polynomials with integer coefficients.

Note that multiplication of two or more variables is not possible in PA. The only type of multiplication that we allow is multiplying a variable by a constant.

### 2.2.2 Extensions of Presburger Arithmetic

While PA is a decidable theory, various extensions of it have been proved undecidable. For example, adding general multiplication to the language leads to undecidability, which follows from Church's negative answer to the *Entscheidungsproblem* [Chu36]. If one instead adds a binary divisibility relation $|$, where $f \mid g$ if and only if $g$ is an integer multiple of $f$, the resulting language is again undecidable as it allows for encoding of general multiplication using only one quantifier alternation [Rob49].

The decision problem for the purely existential subset of PA extended with the multiplication operation is equivalent to Hilbert's Tenth Problem, which is undecidable [Mat70]. However, the decision problem for existential PA extended with divisibility turned out to be decidable [Bel76, Lip76]. A detailed complexity analysis of the latter will be given in Chapter 4.

## 2.3 Linear Algebra

The following result of von zur Gathen and Sieveking [vzGS78] on bounds on the size of generating sets of polyhedral subsets of $\mathbb{Z}^n$ will turn out to be useful to make changes of variables in formulas of Presburger arithmetic and related logics in Chapters 4 and 5. It expresses the solution set of a system of linear equalities and inequalities as a union of linear sets, each of which has a representation of polynomial size. Let $A$ be an $m \times n$ integer matrix of rank $r$ and let $\boldsymbol{b} \in \mathbb{Z}^m$. Let $C$ be a $p \times n$ integer matrix such that the matrix $\binom{A}{C}$ has rank $s$, and let $\boldsymbol{d} \in \mathbb{Z}^p$ . Write $\mu$ for the maximum absolute value of an $(s-1) \times (s-1)$ or $s \times s$ subdeterminant of the matrix $\binom{A\ \boldsymbol{b}}{C\ \boldsymbol{d}}$ that incorporates at least $r$ rows from $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$.

**Theorem 1.** *Given integer matrices $A$ and $C$, integer vectors $\boldsymbol{b}$ and $\boldsymbol{d}$, and $\mu$ defined as above, there exists a finite set $I$ and a collection of $n \times (n-r)$ matrices $E^{(i)}$ and $n \times 1$ vectors $\boldsymbol{u}^{(i)}$, indexed by $i \in I$, all with integer entries bounded by $(n+1)\mu$, such that*

$$\begin{aligned} &\{\boldsymbol{x} \in \mathbb{Z}^n : A\boldsymbol{x} = \boldsymbol{b} \wedge C\boldsymbol{x} \geq \boldsymbol{d}\} \\ &= \bigcup_{i \in I} \{E^{(i)}\boldsymbol{y} + \boldsymbol{u}^{(i)} : \boldsymbol{y} \in \mathbb{Z}^{n-r}, \boldsymbol{y} \geq 0\}\,. \end{aligned}$$

Theorem 1 also gives us the following corollary.

**Corollary 2.** *The decision problem for the purely existential fragment of Presburger arithmetic ($\exists$PA) is in **NP**.*

*Proof.* The problem of finding a solution of a quantifier-free formula of Presburger arithmetic reduces in nondeterministic polynomial time to the problem of finding a solution to a conjunction of linear equalities and inequalities, that is, a system

$$A\boldsymbol{x} = \boldsymbol{b} \wedge C\boldsymbol{x} \geq \boldsymbol{d}$$

as above. By Theorem 1, there exist an $n \times (n-r)$ integer matrix $E$ and an $n \times 1$ integer vector $\boldsymbol{u}$ with all entries bounded by $(n+1)\mu$ such that for all vectors $\boldsymbol{y} \in \mathbb{Z}^{n-r}$ with nonnegative entries, $E\boldsymbol{y} + \boldsymbol{u}$ is a solution to the system. In particular, setting $\boldsymbol{y} = \boldsymbol{0}$, $E\boldsymbol{0} + \boldsymbol{u} = \boldsymbol{u}$ is a solution. So in order to decide in nondeterministic polynomial time if such a system is solvable, we can simply guess an $n \times 1$ vector with integer entries bounded by $(n+1)\mu$ and verify that it is a solution.

□

It is easy to see that the decision problem for ∃PA is also **NP**-hard, since SAT can be encoded in ∃PA, so we see that this problem is **NP**-complete.

## 2.4 Graph Theory

The following basic definitions from graph theory will be necessary in Chapters 5 and 6, that is, the second technical part of this thesis, which is concerned with one-counter automata. There is a close connection between one-counter automata and weighted graphs, which we will explain in more detail in the relevant chapters.

### 2.4.1 Weighted Graphs and Flow Networks

A *weighted graph* is a tuple $G = (V, E, w)$, where $V$ is a finite set of vertices, $E \subseteq V \times V$ is a set of directed edges, and $w : E \to \mathbb{Z}$ assigns an integer weight to each edge. For any edge $(u, v) \in E$, define $\mathrm{start}(u, v) = u$ and $\mathrm{end}(u, v) = v$. For $s, t \in V$, a *path* $\gamma$ from $s$ to $t$ is a word over the alphabet $E$: $\gamma = e_1 e_2 \dots e_n$ with $\mathrm{start}(e_1) = s$, $\mathrm{end}(e_n) = t$, and $\mathrm{end}(e_i) = \mathrm{start}(e_{i+1})$ for all $1 \leq i < n$. The *length* of $\gamma$, denoted by $|\gamma|$, is $n$. The *vertex sequence* of $\gamma$ is $\mathrm{start}(e_1), \mathrm{end}(e_1), \mathrm{end}(e_2), \dots, \mathrm{end}(e_n)$. The *start* of $\gamma$, denoted by $\mathrm{start}(\gamma)$, is $\mathrm{start}(e_1)$. The *end* of $\gamma$, denoted by $\mathrm{end}(\gamma)$, is $\mathrm{end}(e_n)$. A path is *simple* if its vertex sequence contains no repeated vertices. The *weight* of $\gamma$, denoted by $\mathrm{weight}(\gamma)$, is $\sum_{i=1}^{n} w(e_i)$. A *subpath* $\gamma'$ of $\gamma$ is any factor $\gamma' = e_i e_{i+1} \dots e_j$ of $\gamma$. If $\gamma_1$ and $\gamma_2$ are two paths such that $\mathrm{end}(\gamma_1) = \mathrm{start}(\gamma_2)$, then $\gamma_1\gamma_2$ is the concatenation of both paths. A weighted graph is called *strongly connected* if for all pairs of vertices $u, v \in V$, there exist paths in both directions between $u$ and $v$.

A *cycle* $\omega$ is a path such that $\mathrm{start}(\omega) = \mathrm{end}(\omega)$. A cycle is *simple* if there are no repeated vertices except for the starting point, which appears twice. A cycle is *positive* if it has positive weight, *negative* if it has negative weight and *zero-weight* if it has weight zero. We denote by $\omega^k = \underbrace{\omega\omega\dots\omega}_{k \text{ times}}$ the sequence of $k$ iterations of $\omega$.

Given a weighted graph $G = (V, E, w)$ and vertices $s, t \in V$, a *flow* from the *source* $s$ to the *target* $t$ is a function $f : E \to \mathbb{N}$ satisfying the flow conservation condition

$$\sum_{v:(v,u)\in E} f(v,u) = \sum_{v:(u,v)\in E} f(u,v)$$

for all $u \in V \backslash \{s,t\}$. That is, for every vertex other than the source and target, the overall incoming flow is equal to the overall outgoing flow.

The *value* $|f|$ of a given $s$-$t$ flow $f$ is defined as

$$|f| = \sum_{v:(s,v)\in E} f(s,v) - \sum_{v:(v,s)\in E} f(v,s),$$

or equivalently (by flow conservation)

$$|f| = \sum_{v:(v,t)\in E} f(v,t) - \sum_{v:(t,v)\in E} f(t,v).$$

The *weight* of a flow $f$ is

$$\text{weight}(f) = \sum_{e\in E} f(e)w(e).$$

The *support* of a flow $f$ is the set of edges which are assigned a nonzero value by $f$, that is, $\text{support}(f) = \{e \in E : f(e) > 0\}$.

Every $s$-$t$ path $\gamma$ induces an $s$-$t$ flow $f_\gamma$: for each edge $e \in E$, $f_\gamma(e)$ is the number of times that $e$ occurs in $\gamma$. $f_\gamma$ is sometimes called the *Parikh image* of $\gamma$. A flow is called a *path flow* if it corresponds to a path in such a way. It is easy to see, by a variation of the Euler path theorem, that an $s$-$t$ flow $f$ is a path flow if and only if $|f| = 1$ and the subgraph induced by the set of edges $\text{support}(f) \cup \{(t,s)\}$ is strongly connected. If $\gamma = \gamma_1\gamma_2$, then the flow corresponding to $\gamma$ is the sum of the flows corresponding to $\gamma_1$ and $\gamma_2$: $f_\gamma(e) = f_{\gamma_1}(e) + f_{\gamma_2}(e)$ for all $e \in E$.

Given a weighted graph $G$, its *skew transpose* $G^{op}$ is the weighted graph obtained from $G$ by multiplying every weight by $-1$ and then reversing the direction of every edge. Every $s$-$t$ path flow $f$ in $G$ corresponds to a $t$-$s$ path flow $f^{op}$ in $G^{op}$, where $f(u,v) = f^{op}(v,u)$. Notice that $\text{weight}(f^{op}) = -\text{weight}(f)$.

A *parameterised weighted graph* is a tuple $G = (V, E, X, w)$, where $V$ and $E$ are defined as before, $X$ is a finite set of integer variables, and $w : E \to \mathbb{Z} \cup (\mathbb{Z} \times X)$ assigns a weight to each edge which is either an integer or an integer multiple of a variable. Substituting integer values for all the variables in a parameterised weighted graph yields a weighted graph as defined above.

### 2.4.2 The Bellman-Ford Algorithm

The Bellman-Ford algorithm takes as input a weighted graph $G = (V, E, w)$ and a vertex $v \in V$, and computes the length of the shortest path from $v$ to every other vertex in the graph. It also checks for negative cycles in the graph that are reachable from $v$. It is more general than Dijkstra's algorithm, which only works for graphs where all weights are nonnegative, but also has higher complexity, with an upper complexity bound of $O(|V||E|)$ (compare this to the upper bound of $O(|E| + |V| \log |V|)$ for Dijkstra's algorithm).

In the Bellman-Ford algorithm, like in Dijkstra's algorithm, an overestimate of the distance from $v$ to $u$ is stored for each vertex $u$, which is improved iteratively until either the correct distance has been computed for every vertex, or a negative cycle has been detected. Clearly if there is a negative cycle in the graph that can be reached from $v$, there are vertices to which there is no shortest path from $v$. In particular, let $\omega$ be a negative cycle, let $u = \text{start}(\omega)$, and let $\gamma$ be a path from $v$ to $u$. Then $\gamma\omega$ is a "shorter" (of smaller weight) path from $v$ to $u$ than $\gamma$.

The main difference to Dijkstra's algorithm is that while the latter updates only neighbours of the closest vertex to $v$ that has not been processed yet in each iteration, the Bellman-Ford algorithm updates all vertices using all edges in each iteration. This is necessary as for any given vertex $u$, the shortest path from $v$ to $u$ might contain negative weights. There are $|V|$ iterations overall in the algorithm. If any values for the distance of a vertex from $v$ are still improved in the $|V|$-th iteration, a negative cycle has been detected.

The Bellman-Ford algorithm in pseudocode is:

**Input:** Weighted graph $G = (V, E, w)$, source $v \in V$
**Output:**

```
1: for all u ∈ V do
2:    if u = v then
3:       distance(u) ← 0
4:    else
5:       distance(u) ← ∞
6:    end if
7:    predecessor(u) ← null
8: end for
9: for i = 1 to |V| − 1 do
10:   for all e = (u, u′) ∈ E do
```

```
11:       if distance(u) + w(e) < distance(u') then
12:         distance(u') ← distance(u) + w(e)
13:         predecessor(u') ← u
14:       end if
15:     end for
16: end for
17: for all e = (u, u') ∈ E do
18:     if distance(u) + w(e) < distance(u') then
19:       error: negative cycle detected
20:     end if
21: end for
22: return distance
```

Proofs of correctness are given in many standard text, for example [CLRS01, p. 651].

## 2.5 Linear-Time Logic and Büchi Automata

In this section we give definitions and some well-known results concerning linear-time logic and Büchi automata, which will be used in Chapters 5 and 6. For a more comprehensive tutorial including proofs and optimisations, see for example [Wol01].

### 2.5.1 Linear Temporal Logic

*Linear temporal logic* (LTL) is a commonly used logic for reasoning about infinite sequences of states. The set of LTL formulas over a given set of propositions $P$ is given by the grammar

$$\varphi \quad ::= \quad a \quad | \quad \neg\varphi \quad | \quad \varphi \vee \varphi \quad | \quad \mathsf{X}\,\varphi \quad | \quad \varphi \,\mathsf{U}\, \varphi\,,$$

where $a \in P$.

The semantics of LTL are given by a *satisfaction relation* $w, i \vDash \varphi$, specifying when an LTL formula $\varphi$ is satisfied at position $i$ in a word $w \in (2^P)^\omega$, that is, in an infinite

sequence $w$ of subsets of $P$. The satisfaction relation is defined as follows.

$$
\begin{aligned}
w, i \vDash a &\overset{\text{def}}{\Leftrightarrow} a \in w(i) \\
w, i \vDash \neg\varphi &\overset{\text{def}}{\Leftrightarrow} w, i \nvDash \varphi \\
w, i \vDash \varphi_1 \vee \varphi_2 &\overset{\text{def}}{\Leftrightarrow} w, i \vDash \varphi_1 \text{ or } w, i \vDash \varphi_2 \\
w, i \vDash \mathsf{X}\,\varphi &\overset{\text{def}}{\Leftrightarrow} w, i+1 \vDash \varphi \\
w, i \vDash \varphi_1 \,\mathsf{U}\, \varphi_2 &\overset{\text{def}}{\Leftrightarrow} w, j \vDash \varphi_2 \text{ for some } j \geq i \text{ and } w, k \vDash \varphi_1 \text{ for all } i \leq k < j
\end{aligned}
$$

Intuitively, $\mathsf{X}\,\varphi$ means that $\varphi$ will be true in the *next* state of the sequence, and $\varphi_1 \,\mathsf{U}\, \varphi_2$ means that $\varphi_2$ will be true at some point (either in the present state or in the future), and *until* then $\varphi_1$ will be true.

Some commonly used derived operators are:

- $\mathsf{F}\,\varphi \equiv \mathit{true} \,\mathsf{U}\, \varphi$ means that $\varphi$ will *eventually* be true (in the present state or in the future).
- $\mathsf{G}\,\varphi \equiv \neg\,\mathsf{F}\,\neg\varphi$ means that $\varphi$ is *globally* true (that is, in the present state and in every future state).
- $\varphi_1 \,\mathsf{R}\, \varphi_2 \equiv \neg(\neg\varphi_1 \,\mathsf{U}\, \neg\varphi_2)$ means that $\varphi_1$ *releases* $\varphi_2$, that is, if $\varphi_2$ is ever false, then $\varphi_1$ must have been true in some earlier state. The release operator is sometimes included in the grammar of LTL, as it allows the possibility of writing every LTL formula in negation normal form, meaning that negations are only applied to atomic formulas.

We can write the satisfaction relation for the derived operators as

$$
\begin{aligned}
w, i \vDash \mathsf{F}\,\varphi &\Leftrightarrow w, j \vDash \varphi \text{ for some } j \geq i \\
w, i \vDash \mathsf{G}\,\varphi &\Leftrightarrow w, j \vDash \varphi \text{ for all } j \geq i \\
w, i \vDash \varphi_1 \,\mathsf{R}\, \varphi_2 &\Leftrightarrow \text{either: } w, j \vDash \varphi_2 \text{ for all } j \geq i \\
&\qquad\qquad \text{or: } w, j' \vDash \varphi_1 \text{ for some } j' \geq i \text{ and } w, k \vDash \varphi_2 \text{ for all } i \leq k \leq j'
\end{aligned}
$$

We say that a word $w \in (2^P)^\omega$ *satisfies* an LTL formula $\varphi$ if $w, 0 \vDash \varphi$.

### 2.5.2 Büchi Automata

*Büchi automata* are syntactically equivalent to finite automata, but they are different in that they are interpreted over infinite words. Formally, a (nondeterministic) Büchi automaton is a tuple $A = (V, \Sigma, E, V_0, F)$, where

- $V$ is a finite set of states,
- $\Sigma$ is a finite alphabet,

- $E \subseteq V \times \Sigma \times V$ is the transition relation,
- $V_0 \subseteq V$ is the set of initial states,
- $F \subseteq V$ is the set of final (or accepting) states.

A word $w \in \Sigma^\omega$ (i.e., an infinite sequence of symbols from $\Sigma$) is *accepted* by a Büchi automaton $A$ if there is a sequence of states $\lambda : \mathbb{N} \to V$ which maps positions in $w$ to states of $A$ such that the following conditions hold.

- $\lambda(0) \in V_0$.
- For all $i \in \mathbb{N}$, $(\lambda(i), w(i), \lambda(i+1)) \in E$.
- $\inf(\lambda) \cap F \neq \emptyset$, where $\inf(\lambda)$ is the set of states that appear infinitely many times in $\lambda$.

The *language* $L(A)$ accepted by a Büchi automaton $A$ is the set of words accepted by $A$.

A *generalised Büchi automaton* is a tuple $A = (V, \Sigma, E, V_0, \mathcal{F})$, where $V$, $\Sigma$, $E$ and $V_0$ are defined as for Büchi automata and $\mathcal{F} \subseteq 2^V$ is a set of sets of final states. The accepting condition for generalised Büchi automata is identical to the one above for Büchi automata except for the last condition, which becomes

- $\inf(\lambda) \cap F \neq \emptyset$ for all $F \in \mathcal{F}$.

The following lemma shows that Büchi automata and generalised Büchi automata are equivalent models of computation.

**Lemma 3.** *For any generalised Büchi automaton $A$, one can construct in polynomial time a Büchi automaton $A'$ such that $L(A) = L(A')$.*

*Proof.* (Sketch) Given a generalised Büchi automaton $A = (V, \Sigma, E, V_0, (F_1, \ldots, F_k))$, the Büchi automaton $A' = (V', \Sigma, E', V_0', F')$ is defined as follows.

- $V' = V \times \{1, \ldots, k\}$
- $V_0' = V_0 \times \{1\}$
- $((v,i), a, (u,j)) \in E'$ if $(v,a,u) \in E$ and $\begin{cases} j = i & \text{if } v \notin F_i \\ j = (i+1) \bmod k & \text{if } v \in F_i. \end{cases}$
- $F' = F_1 \times \{1\}$

□

### 2.5.3 Translation of LTL Formulas to Büchi Automata

The following theorem shows a fundamental connection between LTL formulas and Büchi automata.

**Theorem 4.** *Let $\varphi$ be an LTL formula over a set of propositions $P$. One can construct in polynomial space a generalised Büchi automaton $A$ such that for any word $w \in (2^P)^\omega$, $w \in L(A)$ if and only if $w, 0 \vDash \varphi$. The size of $A$ is at most singly exponential in the size of $\varphi$.*

The proof of the theorem is by converting $\varphi$ to negation normal form and then considering the set of relevant subformulas of $\varphi$. Subsets of this set are the states of $A$. For details of the proof see [Wol01].

# Chapter 3

# Existential Linear Theory of the $p$-adic Numbers

## 3.1 Introduction

Our first technical contribution concerns the first-order linear theory $L_{LVF}$ (where LVF stands for 'linear valued fields') of the $p$-adic numbers $\mathbb{Q}_p$. We consider the complexity of the decision problem for existential sentences in this theory, with the prime $p$ also regarded as part of the input. Nearly three decades ago, Weispfenning showed that this problem lies between **NP** and **EXPTIME**, and raised the question of its precise complexity as an open problem [Wei88]. In this chapter, we show that the problem lies in the Counting Hierarchy **CH**, and thus within **PSPACE**.

The proof involves a careful analysis of the quantifier elimination procedure for the existential linear theory of $\mathbb{Q}_p$, following Cohen [Coh69]. The **CH** bound enters through the need to check exact divisibility for pairs of integers represented succinctly as arithmetic circuits, for which we use results of [ABD14, ABH01].

Besides improving the best known upper complexity bound for an open problem, which is an interesting result in its own right, our results from this chapter will also prove to be an important building block for the proof of the main result of Chapter 4. This is due to a connection between the $p$-adic valuation function which is part of the language $L_{LVF}$, and the divisibility relation which is included in the language PAD (Presburger arithmetic with divisibility).

## 3.2 Preliminaries

### 3.2.1 The Field of $p$-adic Numbers

For a given prime $p$, the *$p$-adic valuation* (or *$p$-adic order*) $v_p : \mathbb{Q} \to \mathbb{Z} \cup \{\infty\}$ is defined as follows.

$$v_p(a) = \begin{cases} \max\{k \in \mathbb{N} : p^k \mid a\} & \text{if } a \in \mathbb{Z} \setminus \{0\} \\ \infty & \text{if } a = 0 \\ v_p(b) - v_p(c) & \text{if } a = \frac{b}{c}. \end{cases}$$

The following simple examples illustrate the intuition behind this definition: the $p$-adic valuation of a rational number is the exponent in the biggest power of $p$ that can be multiplied by an integer to yield that number.

**Example 5.**

- $v_2(8) = 3$ *since* $2^3 \mid 8$ *and* $2^4 \nmid 8$.
- $v_3(6) = 1$ *since* $3^1 \mid 6$ *and* $3^2 \nmid 6$.
- $v_p(1) = 0$ *for all primes* $p$.
- $v_p(p) = 1$ *for all primes* $p$.
- $v_3(\frac{1}{3}) = v_3(1) - v_3(3) = 0 - 1 = -1$.
- $v_3(\frac{5}{27}) = v_3(5) - v_3(27) = 0 - 3 = -3$.
- $v_3(2^4 \times 3^6 \times 5^2 \times 7) = 6$.

Two key properties of $v_p$ are the *homomorphism property*

$$v_p(xy) = v_p(x) + v_p(y) \tag{3.1}$$

and the (non-Archimedean) *triangle inequality*

$$v_p(x + y) \geq \min(v_p(x), v_p(y)) \tag{3.2}$$

with the special equality case

$$v_p(x + y) = \min(v_p(x), v_p(y)) \text{ if } v_p(x) \neq v_p(y). \tag{3.3}$$

Here we adopt the convention that $n < \infty$ for all $n \in \mathbb{Z}$ and $n + \infty = \infty$ for all $n \in \mathbb{Z} \cup \{\infty\}$. Note also that $v_p(a) = v_p(-a)$ for all $a$.

The *$p$-adic absolute value* (or *$p$-adic norm*) $|\cdot|_p : \mathbb{Q} \to \mathbb{Q}$ is defined as

$$|a|_p = \begin{cases} p^{-v_p(a)} & \text{if } a \in \mathbb{Q} \setminus \{0\} \\ 0 & \text{if } a = 0. \end{cases}$$

Intuitively, the $p$-adic absolute value of a rational number is the inverse of the biggest power of $p$ that can be multiplied by an integer to yield that number.

**Example 6.**

- $|8|_2 = 2^{-v_2(8)} = 2^{-3} = \frac{1}{8}$.
- $|24|_2 = 2^{-v_2(24)} = 2^{-3} = \frac{1}{8}$.
- $|1|_p = p^{-v_p(1)} = p^0 = 1$ *for all primes* $p$.
- $|\frac{5}{27}|_3 = 3^{-v_3(\frac{5}{27})} = 3^3 = 9$.
- $|2^4 \times 3^6 \times 5^2 \times 7|_3 = \frac{1}{3^6}$.

The field $\mathbb{Q}_p$ of *p-adic numbers* is the Cauchy completion of the field $\mathbb{Q}$ of rational numbers with respect to the metric $d(a, b) = |a - b|_p$. Compare this definition with the real numbers, which are the Cauchy completion of the rational numbers with respect to the metric $d'(a, b) = |a - b|$. The definition of the $p$-adic numbers is completely analogous to that of the real numbers, the only difference being the different definition of absolute value. Intuitively, this means a different interpretation of "size" of a number and "distance" between two numbers: in the $p$-adic number system, higher powers of $p$ are considered "smaller" than lower powers of $p$, and two numbers are close to each other if their difference is small in that way. For example, a simple Cauchy sequence in the $p$-adic numbers is

$$x_n = p^n,$$

which converges to zero with respect to the norm $|\cdot|_p$.

Any $p$-adic number $a \in \mathbb{Q}_p \setminus \{0\}$ can be expressed as a *p-adic expansion*, that is, as a one-way infinite power series

$$a = \sum_{i=k}^{\infty} a_i p^i \tag{3.4}$$

where $k \in \mathbb{Z}$, $a_i \in \{0, \ldots, p-1\}$ for each $i$, and $a_k \neq 0$. This sum converges with respect to the norm $|\cdot|_p$. Note that $p$-adic expansions extend infinitely to large values of $p$. This should be contrasted with the base $p$ expansion of a real number $a$,

$$a = \pm \sum_{i=-\infty}^{k} a_i p^i \tag{3.5}$$

which converges with respect to the norm $|\cdot|$ and extends infinitely to small values of $p$.

The $p$-adic valuation extends to a map $v_p : \mathbb{Q}_p \to \mathbb{Z} \cup \{\infty\}$ with $v_p(a) = k$ for $a$ as in (3.4). The $p$-adic absolute value then extends to a map $|\cdot|_p : \mathbb{Q}_p \to \mathbb{Q}$ in the obvious way. Both the homomorphism property and the triangle inequality still

hold for the extended version of the $p$-adic valuation. The set of *$p$-adic integers* $\mathbb{Z}_p = \{x \in \mathbb{Q}_p : v_p(x) \geq 0\}$ forms a subring of $\mathbb{Q}_p$ that contains $\mathbb{Z}$. We will sometimes call $\mathbb{Q}_p$ the *$p$-adic rationals* to highlight the difference from the $p$-adic integers.

**Example 7.** *Let $p = 3$ and consider the sequence*

$$\begin{aligned}
3 - 1 &= 2 &&= 2 \times 1 \\
3^2 - 1 &= 8 &&= 2 \times 3 + 2 \times 1 \\
3^3 - 1 &= 26 &&= 2 \times 3^2 + 2 \times 3 + 2 \times 1 \\
3^4 - 1 &= 80 &&= 2 \times 3^3 + 2 \times 3^2 + 2 \times 3 + 2 \times 1 \\
&\ldots
\end{aligned}$$

*Since in the 3-adic number system, the sequence $3, 3^2, 3^3, 3^4, \ldots$ converges to zero, the above sequence converges to $-1$. Hence the 3-adic expansion of $-1$ is*

$$-1 = \sum_{i=0}^{\infty} 2 \times 3^i = \ldots 22222_3.$$

*Now consider the sequence*

$$\begin{aligned}
\frac{3^2 - 1}{4} &= 2 &&= 2 \times 1 \\
\frac{3^4 - 1}{4} &= 20 &&= 2 \times 3^2 + 0 \times 3 + 2 \times 1 \\
\frac{3^6 - 1}{4} &= 182 &&= 2 \times 3^4 + 0 \times 3^3 + 2 \times 3^2 + 0 \times 3 + 2 \times 1 \\
&\ldots
\end{aligned}$$

*which converges to $-\frac{1}{4}$ in the 3-adics, which gives us the 3-adic expansions*

$$-\frac{1}{4} = \sum_{i=0}^{\infty} 2 \times 3^{2i} = \ldots 20202_3$$

*and*

$$\frac{3}{4} = -\frac{1}{4} + 1 = \left( \sum_{i=1}^{\infty} 2 \times 3^{2i} \right) + 3 = \ldots 2020210_3$$

*Note that $v_3(-\frac{1}{4}) = 0 \geq 0$ and $v_3(\frac{3}{4}) = 1 \geq 0$, so $-\frac{1}{4}$ and $\frac{3}{4}$ are examples of 3-adic integers which are not integers.*

The following theorem shows a connection between $p$-adic valuations and the divisibility relation over the integers.

**Theorem 8.** *If $a, b \in \mathbb{Z}$ then $a \mid b$ if and only if $v_p(a) \leq v_p(b)$ for all primes $p$.*

*Proof.* Suppose that $a$ and $b$ are integers such that $a \mid b$, that is, there exists an integer $c$ with $ca = b$. Then for any prime $p$,

$$\begin{aligned} v_p(b) &= v_p(ca) && \{\text{since } ca = b\} \\ &= v_p(c) + v_p(a) && \{\text{by the homomorphism property}\} \\ &\geq v_p(a) && \{\text{since } c \text{ is a } p\text{-adic integer}\} \end{aligned}$$

Conversely, suppose that $a$ and $b$ are integers such that $v_p(a) \leq v_p(b)$ for all primes $p$. Let the prime factorisation of $b$ be $p_1^{k_1} p_2^{k_2} \dots p_m^{k_m}$. For all $1 \leq i \leq m$, we know that $v_{p_i}(a) \leq v_{p_i}(b)$, that is, the exponent of $p_i$ in the prime factorisation of $a$ can be at most $v_{p_i}(b) = k_i$. For any prime $p$ such that $p \neq p_i$ for all $1 \leq i \leq m$, we know that $v_p(a) \leq v_p(b) = 0$. That is, $p$ does not occur (or occurs only with exponent zero) in the prime factorisation of $a$. It clearly follows that $b$ is an integer multiple of $a$. □

The interested reader can find a more comprehensive introduction to the $p$-adic numbers and $p$-adic analysis in [Bak11], but most of the topics covered there are beyond the scope of this thesis.

### 3.2.2 The Counting Hierarchy

An *arithmetic circuit (or straight-line program)* is a finite directed acyclic graph, with input nodes of indegree 0 labelled with constants 0 or 1, and with internal nodes labelled either $+$, $\times$, or $-$. Such a circuit has a distinguished output node of outdegree 0 that represents an integer in the obvious way. The *size* of a circuit is its number of nodes, and the *depth* of a circuit is the length of the longest path from an input node to the output node. Note that we only consider arithmetic circuits with input values in $\{0, 1\}$ in this thesis, while more general definitions (allowing integers and variables as inputs) exist in the literature.

Representations of integers as arithmetic circuits can be very succinct. For example, Figure 3.1 shows a circuit of size $n + 3$ that represents the number $2^{2^n}$, for any given $n \in \mathbb{N}$. The idea is to use iterated squaring via a sequence of $n$ product nodes. In general, a circuit with $O(n)$ vertices can represent a number with up to $O(2^n)$ bits.

Recall that the Polynomial Hierarchy is defined to be

$$\mathbf{PH} = \bigcup_{n \geq 0} \Delta_n^P = \bigcup_{n \geq 0} \Sigma_n^P = \bigcup_{n \geq 0} \Pi_n^P$$

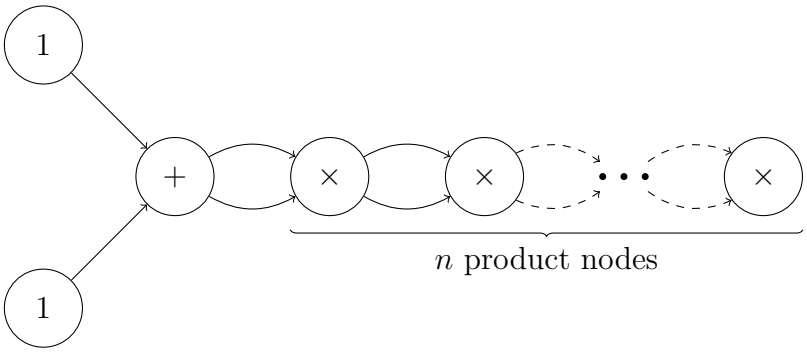


Figure 3.1: The number $2^{2^n}$ represented by an arithmetic circuit with two input nodes, one sum node and $n$ product nodes.

with

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathbf{P}$$

$$\Delta_{n+1}^P = \mathbf{P}^{\Sigma_n^P}, \quad \Sigma_{n+1}^P = \mathbf{NP}^{\Sigma_n^P}, \quad \Pi_{n+1}^P = \mathbf{coNP}^{\Sigma_n^P}.$$

The notation $A^B$, for complexity classes $A$ and $B$, means the complexity class of decision problems solvable by an algorithm in class $A$ with an oracle for some complete problem in class $B$. Recall also that **PP** is the class of languages $L$ for which there is a nondeterministic polynomial-time machine such that for each word $x$, $x \in L$ if and only if a strict majority of the computation paths on input $x$ end in an accepting state. The abbreviation **PP** stands for probabilistic polynomial time.

The *Counting Hierarchy* [Tor91, Wag86] is the complexity class which is defined analogously to the Polynomial Hierarchy in terms of **PP** instead of **NP**:

$$\mathbf{CH} = \bigcup_{n \geq 0} \mathbf{CH}_n$$

where

$$\mathbf{CH}_0 = \mathbf{PP}, \quad \mathbf{CH}_{n+1} = \mathbf{PP}^{\mathbf{CH}_n}.$$

It is straightforward that $\mathbf{CH} \subseteq \mathbf{PSPACE}$. Toda's Theorem [Tod91] states that any problem in the Polynomial Hierarchy **PH** is polynomial-time Turing reducible to a problem in **PP**, i.e., $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{PP}}$, so we see that $\mathbf{PH} \subseteq \mathbf{CH} \subseteq \mathbf{PSPACE}$. In [ABD14, ABKM09] some natural decision problems associated with arithmetic circuits are shown to belong to the Counting Hierarchy.

## 3.3 Syntax

Fix a prime $p$. Following Cohen [Coh69], we work with a two-sorted first-order theory $L_{LVF}$ for linear valued fields. There is a sort for the set of $p$-adic numbers $\mathbb{Q}_p$ and a

sort for the set of *values* $\mathbb{Z} \cup \{\infty\}$. We write $x_1, x_2, \ldots$ for variables of $p$-adic sort and $u_1, u_2, \ldots$ for variables of value sort.

We are interested in the *linear* theory of $\mathbb{Q}_p$, so $L_{LVF}$ includes a constant symbol of $p$-adic sort for each element of $\mathbb{Q}$, a constant symbol of value sort for each element of $\mathbb{Z}$, and a binary function symbol $+$ on both the $p$-adic and value sorts. We also have a binary order relation $<$ on the value sort, and a unary function symbol $v$ from the $p$-adic sort to the value sort denoting the $p$-adic valuation $v_p$. We will consider the *existential subset* of $L_{LVF}$, that is, we do not allow any multiplication of or universal quantification over variables of either sort.

It is technically convenient to restrict the range of the variables $u_1, u_2, \ldots$ to be $\mathbb{Z}$, that is, to exclude $\infty$. This restriction is without loss of generality, since $\infty$ is denoted by $v(0)$ and can be treated as a special case in each formula. Moreover, as a consequence of the restriction, any existential $L_{LVF}$-formula that does not contain a term of $p$-adic sort can be considered as a formula of existential Presburger arithmetic. Henceforth we will refer to the $u_i$ as *integer variables*.

Weispfenning [Wei88] considers the existential first-order linear theory of $\mathbb{Q}_p$ in a single-sorted formalism in which the binary $p$-adic divisibility relation $v(a) \leq v(b)$ is taken as primitive rather than the $p$-adic valuation $v$. However it is straightforward to translate from the one-sorted to the two-sorted setting.

## 3.4 Quantifier Elimination

Quantifier elimination for the existential linear theory of $\mathbb{Q}_p$ has been studied in [Stu00, Wei88]. In particular, [Wei88] uses quantifier elimination to show that the truth of an existential sentence $\varphi$ with $n$ variables can be decided in time $M^{O(n)}$, where $M$ is the total size of $\varphi$ and $p$ when integers are represented in binary. Below we give a variant of the elimination procedure which is instrumental in obtaining our Counting-Hierarchy bound for the decision problem (see the final paragraph of the proof of Proposition 13 in Section 3.5).

Again we assume that the prime $p$ is fixed. As a preliminary step we prove the following lemma.

**Lemma 9.** *The decision problem for existential $L_{LVF}$ can be reduced to the special case of formulas of the form*

$$\exists u_1 \ldots \exists u_m \exists x_1 \ldots \exists x_n \varphi \,, \tag{3.6}$$

*where the $u_i$ range over $\mathbb{Z}$, the $x_i$ range over $\mathbb{Q}_p$, and $\varphi$ is a quantifier-free conjunction of atomic formulas of the form $v(f) = s$, $s < t$, and $s = t$, where $s$ and $t$ are linear terms over integer variables and $f$ is a linear term over variables of $p$-adic sort. Moreover, each term $v(f)$ may only occur at most once.*

*Proof.* Given an arbitrary purely existential formula $\psi$ of $L_{LVF}$, we first write it in prenex normal form, moving all existential quantifiers to the beginning of the formula. All disjunctions can be moved left past the existential quantifiers, which gives us a disjunction of formulas of the form $\exists u_1 \dots \exists u_m \exists x_1 \dots \exists x_n \psi'$, where $\psi'$ is a quantifier-free conjunction of atomic formulas and negated atomic formulas.

We aim to simplify $\psi'$ so that the valuation $v$ is only mentioned in atoms of the form $v(f) = s$ for a $p$-adic term $f$ and an integer linear term $s$. To do this we perform a case analysis on whether or not each sub-term $v(f)$ that violates the condition is equal to $\infty$, by writing $\psi'$ as a disjunction which includes one disjunct for each case. For the case $v(f) \neq \infty$ we add an equation $v(f) = u$, where $u$ is a fresh existentially quantified integer variable, and replace all other occurences of $v(f)$ by $u$. In the case that $v(f) = \infty$ we add an equation $f = 0$ and rewrite all atoms involving $v(f)$ either to true or false, as dictated by arithmetic and order properties of $\infty$. After this simplification, all terms of value sort denote integers. If for any $p$-adic term $f$, there are formulas $v(f) = s$ and $v(f) = s'$, we replace the second formula by $s = s'$.

We can then rewrite each disequality $f \neq g$ between terms of $p$-adic sort as $v(f - g) = u$, for $u$ a fresh existentially quantified integer variable. Next we collect all equalities between terms of $p$-adic sort into a system of linear equalities $A\boldsymbol{x} = \boldsymbol{b}$ with rational coefficients. If this system has a solution then it has one of the form $\boldsymbol{x} = E\boldsymbol{y} + \boldsymbol{c}$, where $E$ is a matrix of rational numbers, $\boldsymbol{c}$ a vector of rational numbers, and $\boldsymbol{y}$ a vector of fresh variables. We can then eliminate the equalities $A\boldsymbol{x} = \boldsymbol{b}$ by substituting $E\boldsymbol{y} + \boldsymbol{c}$ for $\boldsymbol{x}$ in all subformulas of the form $v(f) = u$. Note that the size of $E$ and $\boldsymbol{c}$ is polynomial in the size of $A$ and $\boldsymbol{b}$, and both can be computed by solving the system of linear equalities, for example using Gaussian elimination.

Finally, any disequality $s \neq t$ between terms of value sort can be rewritten as $s < t \vee t < s$. Again we move disjunctions to the beginning of the formula, and we end up with a disjunction of formulas of type (3.6).

□

**Example 10.** *Consider the formula*

$$\begin{aligned}\varphi \equiv\ \exists u_1 \exists x_1 \exists x_2 \exists x_3 \exists x_4\, & v(x_1 + 2x_2 - x_3 - 3x_4) = v(4x_1 - x_2 + x_3 + 2) \\ & \wedge v(-x_1 + x_2 - x_3 - x_4) = 3u_1 + 2 \\ & \wedge v(x_1 - 2x_2 + x_3 - 3x_4) = -u_1.\end{aligned}$$

*The second and third conjunct are already in the form $v(f) = s$. Depending on the values of the terms in the first conjunct, we split the formula into four disjuncts:*

- *For the case that $v(x_1 + 2x_2 - x_3 - 3x_4) \neq \infty$ and $v(4x_1 - x_2 + x_3 + 2) \neq \infty$ we write*

$$\begin{aligned}\varphi_1 \equiv\ \exists u_1 \exists u_2 \exists x_1 \exists x_2 \exists x_3 \exists x_4\, & v(x_1 + 2x_2 - x_3 - 3x_4) = u_2 \\ & \wedge v(4x_1 - x_2 + x_3 + 2) = u_2 \\ & \wedge v(-x_1 + x_2 - x_3 - x_4) = 3u_1 + 2 \\ & \wedge v(x_1 - 2x_2 + x_3 - 3x_4) = -u_1.\end{aligned}$$

- *For the case that $v(x_1 + 2x_2 - x_3 - 3x_4) \neq \infty$ and $v(4x_1 - x_2 + x_3 + 2) = \infty$ we write*

$$\begin{aligned}\varphi_2 \equiv\ \exists u_1 \exists u_2 \exists x_1 \exists x_2 \exists x_3 \exists x_4\, & v(x_1 + 2x_2 - x_3 - 3x_4) = u_2 \\ & \wedge 4x_1 - x_2 + x_3 + 2 = 0 \\ & \wedge \text{“}u_2 = \infty\text{”} \\ & \wedge v(-x_1 + x_2 - x_3 - x_4) = 3u_1 + 2 \\ & \wedge v(x_1 - 2x_2 + x_3 - 3x_4) = -u_1.\end{aligned}$$

  *The “$u_3 = \infty$” part is rewritten to $false$, so $\varphi_2$ evaluates to $false$.*
- *Similarly, for the case that $v(x_1+2x_2-x_3-3x_4) = \infty$ and $v(4x_1-x_2+x_3+2) \neq \infty$, the corresponding disjunct $\varphi_3$ also evaluates to $false$.*
- *Finally we have the case where $v(x_1+2x_2-x_3-3x_4) = (4x_1-x_2+x_3+2) = \infty$. This gives us the system of linear equalities*

$$\begin{aligned} x_1 \ &+ 2x_2 - x_3 - 3x_4 = 0 \\ 4x_1 - x_2 \ &+ x_3 \qquad\quad = -2 \end{aligned}$$

  *One way of solving this system would give us $x_1 = y_1$, $x_2 = -5y_1 + 3y_2 - 2$, $x_3 = -9y_1 + 3y_2 - 4$, $x_4 = y_2$, for new existentially quantified p-adic variables $y_1$ and $y_2$. Substituting this change of variables gives us*

$$\begin{aligned}\varphi_4 \equiv\ \exists u_1 \exists y_1 \exists y_2\, & v(3y_1 - y_2 + 2) = 3u_1 + 2 \\ & \wedge v(2y_1 - 6y_2) = -u_1.\end{aligned}$$

*We then write $\varphi$ as the equivalent disjunction $\varphi_1 \vee \varphi_4$.*

We will show how to eliminate the quantifiers over the $p$-adic variables $x_1, \ldots, x_n$ in (3.6) from the inside out, possibly adding new existentially quantified integer variables. In the end we obtain an equivalent formula of Presburger arithmetic.

It suffices to show how to eliminate a single existential quantifier over a $p$-adic variable. To this end, we consider for a given $p$-adic variable $y$ (with $y = x_i$ for some $i$) the formula with one existential quantifier over $y$ which includes all conjuncts from the matrix of (3.6) that mention $y$. Such a formula is of the form

$$\exists y \bigwedge_{i \in I} v(a_i y - f_i) = s_i \,, \tag{3.7}$$

where $I$ is a finite index set such that for each $i \in I$, $a_i \in \mathbb{Q} \setminus \{0\}$, $f_i$ is a $p$-adic term which does not mention $y$, and $s_i$ is an integer term.

For each $i \in I$, by the homomorphism property (3.1) of valuations we have $v(a_i y - f_i) = v(a_i(y - g_i)) = v(a_i) + v(y - g_i)$, where $g_i := \frac{1}{a_i} f_i$. Thus we can equivalently rewrite (3.7) as

$$\exists y \bigwedge_{i \in I} v(y - g_i) = t_i \,, \tag{3.8}$$

where $t_i := s_i - v(a_i)$ for each $i \in I$.

Let $\mathcal{I}$ be the set of all possible partitions of the form $(I_0, \ldots, I_q)$ of $I$, with $1 \leq q < p$, $I_0$ possibly empty and $I_j$ non-empty for $1 \leq j \leq q$. For a given partition $(I_0, \ldots, I_q)$ we define $\mu_j = \min(I_j)$ for $1 \leq j \leq q$. We think of the $\mu_j$ as representative elements of the sets that they belong to. The following lemma shows a way to rewrite (3.8) as an equivalent disjunction of formulas which do not mention $y$. Each disjunct corresponds to a partition of the index set $I$.

**Lemma 11.** *Let $\varphi = \exists y \bigwedge_{i \in I} v(y - g_i) = t_i$. Then $\varphi$ is equivalent to the disjunction $\bigvee_{(I_0, \ldots, I_q) \in \mathcal{I}} \psi_{I_0, \ldots, I_q}$, where for each partition $(I_0, \ldots, I_q)$ of $I$, writing $\mu_j$ for the representative elements as above the formula $\psi_{I_0, \ldots, I_q}$ is defined as*

$$\begin{aligned} \psi_{I_0, \ldots, I_q} \stackrel{\text{def}}{=} & \bigwedge_{i \in I_0} t_i < t_{\mu_1} \wedge \bigwedge_{i \in I \setminus I_0} t_i = t_{\mu_1} \\ & \wedge \bigwedge_{j=2}^{q} v(g_{\mu_j} - g_{\mu_1}) = t_{\mu_1} \wedge \bigwedge_{i \in I_0} v(g_i - g_{\mu_1}) = t_i \\ & \wedge \bigwedge_{j=1}^{q} \bigwedge_{i \in I_j \setminus \{\mu_j\}} v(g_i - g_{\mu_j}) > t_{\mu_1} \,. \end{aligned} \tag{3.9}$$

*Proof.* In the first part of the proof, we will show that for any given satisfying assignment of $\varphi$, we can construct a partition $(I_0, \ldots, I_q) \in \mathcal{I}$ based on the assignment such that the same assignment also satisfies $\psi_{I_0,\ldots,I_q}$. In the second part of the proof, we will show that for any partition $(I_0, \ldots, I_q) \in \mathcal{I}$, if the formula $\psi_{I_0,\ldots,I_q}$ has a satisfying assignment then we can extract a satisfying assignment of $\varphi$ from this partition and assignment.

Consider a satisfying assignment $\nu$ (for all variables, of both $p$-adic and integer sort, and including $y$) of the conjunction $\bigwedge_{i\in I} v(y - g_i) = t_i$ which forms the matrix of the formula (3.8). That is, $v(y(\nu) - g_i(\nu)) = t_i(\nu)$ holds for all $i \in I$. Intuitively, we want to assign each $i \in I$ to a block $I_j$, $0 \leq j \leq q$ depending on the $p$-adic expansion of $y(\nu) - g_i(\nu)$. The value of $q$ will also depend on these $p$-adic expansions.

Formally, we write $\mu_1 = \arg\max\{t_i(\nu) : i \in I\}$, that is, the maximal value of all the $p$-adic valuations of $y(\nu) - g_i(\nu)$ for $i \in I$ is $t_{\mu_1}(\nu)$. We also write $I_0 = \{i \in I : t_i(\nu) < t_{\mu_1}(\nu)\}$. So $I_0$ is the set which contains all values $i \in I$ such that $v(y(\nu) - g_i(\nu))$ is not maximal. Now

- for all $i \in I_0$, the $p$-adic expansion of $y(\nu) - g_i(\nu)$ has a nonzero entry somewhere up to the $(t_{\mu_1}(\nu))$-th $(p^{t_{\mu_1}(\nu)-1})$ digit, and
- for all $i \in I \setminus I_0$, the $p$-adic expansion of $y(\nu) - g_i(\nu)$ has all entries up to the $(t_{\mu_1}(\nu))$-th digit equal to zero, and the $(t_{\mu_1}(\nu) + 1)$-th digit unequal to zero.

We then partition the set $I \setminus I_0 = \{i \in I : t_i(\nu) = t_{\mu_1}(\nu)\}$ into blocks $I_1, \ldots, I_q$, such that $i$ and $j$ lie in the same block if and only if the $(t_{\mu_1}(\nu) + 1)$-th digits of $y(\nu) - g_i(\nu)$ and $y(\nu) - g_j(\nu)$ are identical. Equivalently, $i$ and $j$ lie in the same block if $v(g_i(\nu) - g_j(\nu)) > t_{\mu_1}(\nu)$, and they lie in different blocks if $(v(g_i(\nu) - g_j(\nu)) = t_{\mu_1}(\nu)$. To understand why the two conditions are equivalent, note that

$$\begin{aligned} v(g_i(\nu) - g_j(\nu)) &= v((y(\nu) - g_j(\nu)) - (y(\nu) - g_i(\nu))) \\ &\geq \min(v(y(\nu) - g_j(\nu)), v(y(\nu) - g_i(\nu))) \\ &= t_{\mu_1}(\nu) \end{aligned}$$

by the triangle inequality (3.2) and since $v(y(\nu) - g_j(\nu)) = v(y(\nu) - g_i(\nu)) = t_{\mu_1}(\nu)$. Now

- if the $(t_{\mu_1}(\nu) + 1)$-th digits of $y(\nu) - g_i(\nu)$ and $y(\nu) - g_j(\nu)$ are different, their difference $v(g_i(\nu) - g_j(\nu))$ again has all entries up to the $(t_{\mu_1}(\nu))$-th digit equal to zero, and the $(t_{\mu_1}(\nu) + 1)$-th digit unequal to zero, so its $p$-adic valuation is again $t_{\mu_1}(\nu)$, that is, $v(g_i(\nu) - g_j(\nu)) = t_{\mu_1}(\nu)$.

- if the $(t_{\mu_1}(\nu)+1)$-th digits of $y(\nu)-g_i(\nu)$ and $y(\nu)-g_j(\nu)$ are identical, their difference $v(g_i(\nu)-g_j(\nu))$ has has all entries up to the $(t_{\mu_1}(\nu)+1)$-th digit equal to zero, so its $p$-adic valuation is larger than $t_{\mu_1}(\nu)$, that is, $v(g_i(\nu)-g_j(\nu)) > t_{\mu_1}(\nu)$.

It is not difficult to see that this definition of the partition of $I$ is sound, that is, that $\{(i,j) : v(g_i(\nu)-g_j(\nu)) > t_{\mu_1}(\nu)\}$ is an equivalence relation. We can also ensure that $\mu_1$ is consistent with the above definition of $\mu_j = \min(I_j)$ by simply making sure that $\mu_1 \in I_1$. From the condition defining the sets $I_1, \ldots, I_q$ in terms of $p$-adic expansions, it is clear that we must have $q < p$. We must now show that for the chosen partition $\{I_0, \ldots, I_q\}$, $\nu$ satisfies the formula $\psi_{I_0,\ldots,I_q}$.

That $\nu$ satisfies the first, second, fourth, and fifth conjuncts of $\psi_{I_0,\ldots,I_q}$ directly follows from the choice of the sets $I_0, \ldots, I_q$ and corresponding representatives $\mu_1, \ldots, \mu_q$. That $\nu$ satisfies the fourth conjunct in $\psi_{I_0,\ldots,I_q}$ follows from the second part of the triangle inequality, (3.3), since for $i \in I_0$ we have $t_i(\nu) < t_{\mu_1}(\nu)$ and hence

$$\begin{aligned} v(g_i(\nu) - g_{\mu_1}(\nu)) &= \min(v(y(\nu) - g_{\mu_1}(\nu)), v(y(\nu) - g_i(\nu))) \\ &= \min(t_{\mu_1}(\nu), t_i(\nu)) \\ &= t_i(\nu)\,. \end{aligned}$$

Conversely, for the second part of the proof we show that an assignment $\nu$ that satisfies $\psi_{I_0,\ldots,I_q}$ for some arbitrary partition $(I_0, \ldots, I_q)$ of $I$, with $q < p$ and $I_1 \ldots, I_q$ non-empty, also satisfies the formula (3.8) for some value of $y$. In this case, we know from the third conjunct of $\psi_{I_0,\ldots,I_q}$ that $v(g_{\mu_j}(\nu) - g_{\mu_1}(\nu)) = t_{\mu_1}(\nu)$ for $j = 2, \ldots, q$, meaning that the $p$-adic expansion of $g_{\mu_j}(\nu)$ is identical to the one of $g_{\mu_1}(\nu)$ up to position $t_{\mu_1}(\nu)$ and differs in position $t_{\mu_1}(\nu)+1$ for all $j \in \{2, \ldots, q\}$, and we also know that $q < p$. So we may choose a value $a$ for $y$ such that $v(a - g_{\mu_j}(\nu)) = t_{\mu_1}(\nu)$ for $j = 1, \ldots, q$: simply choose the first $t_{\mu_1}(\nu)$ digits of the $p$-adic expansion of $a$ to be the same as in those of $g_{\mu_1}(\nu)$, and choose the digit in position $t_{\mu_1}(\nu)+1$ to be different to that in the expansion of each of the $g_{\mu_j}(\nu)$, $j \in \{1, \ldots, q\}$. All other digits of $a$ can be chosen arbitrarily. We claim that this choice of a value for $y$ satisfies $\bigwedge_{i \in I} v(y - g_i(\nu)) = t_i(\nu)$. Indeed for $i \in I_0$, since $t_i(\nu) < t_{\mu_1}(\nu)$, by (3.3) and the fourth conjunct of $\psi_{I_0,\ldots,I_q}$ we have

$$\begin{aligned} v(a - g_i(\nu)) &= \min(v(a - g_{\mu_1}(\nu)), v(g_{\mu_1}(\nu) - g_i(\nu))) \\ &= \min(t_{\mu_1}(\nu), t_i(\nu)) \\ &= t_i(\nu)\,, \end{aligned}$$

and for $j \in \{1, \ldots, q\}$ and $i \in I_j$, by (3.3) and the second and fifth conjuncts of $\psi_{I_0,\ldots,I_q}$ we have

$$v(a - g_i(\nu)) = \min(\underbrace{v(a - g_{\mu_j}(\nu))}_{=t_{\mu_1}(\nu)}, \underbrace{v(g_{\mu_j}(\nu) - g_i(\nu))}_{>t_{\mu_1}(\nu)}) = t_{\mu_1}(\nu) = t_i(\nu)\,.$$

□

Given any formula of the form (3.8), we can essentially apply Lemma 11 repeatedly to eliminate all the quantifiers over variables of $p$-adic sort. One small caveat is that (3.9) includes conjuncts of the form $v(f) > s$ for a $p$-adic term $f$ and an integer term $s$, which are not allowed in formulas of the form (3.8). To maintain the condition that all atomic formulas mentioning the valuation $v$ have the form $v(f) = s$ for an integer term $s$, we apply Lemma 9 once more after each elimination of a quantifier, introducing new existentially quantified integer variables in the process. When eventually only one formula $v(f) = s$ is left, clearly this conjunct has a solution for the variables in $f$ given any value for $s$.

**Example 12.** *Let $p = 3$. Consider from Example 10 the formula $\varphi_4$. The matrix of this formula is*

$$\begin{aligned} & v(3y_1 - y_2 + 2) = 3u_1 + 2 \\ \wedge\, & v(2y_1 - 6y_2) = -u_1. \end{aligned}$$

*We aim to eliminate the $p$-adic variable $y_2$. To this end, since $v_3(6y_2 - 2y_1) = v_3(6) + v_3(y_2 - \frac{1}{3}y_1) = 1 + v_3(y_2 - \frac{1}{3}y_1)$ we rewrite the two conjuncts as*

$$\begin{aligned} & v(y_2 - (3y_1 + 2)) = 3u_1 + 2 \\ \wedge\, & v(y_2 - \frac{1}{3}y_1) = -u_1 - 1 \end{aligned}$$

*or equivalently,*

$$\bigwedge_{i=1}^{2} v(y_2 - g_i) = t_i$$

*for $g_1 = 3y_1 + 2$, $g_2 = \frac{1}{3}y_1$, $t_1 = 3u_1 + 2$, and $t_2 = -u_1 - 1$. There are three ways to partition the set $\{1, 2\}$ into sets $I_0$ and $I_1$ given our condition that $I_0$ may be empty but $I_1$ may not:*

- *$I_0 = \{1\}, I_1 = \{2\}$. Then $\mu_1 = 2$ and*

$$\begin{aligned} \psi_{I_0,I_1} &\equiv t_1 < t_2 \wedge v(g_1 - g_2) = t_1 \\ &\equiv 3u_1 + 2 < -u_1 - 1 \wedge v(3y_1 + 2 - \frac{1}{3}y_1) = 3u_1 + 2 \\ &\equiv 4u_1 + 3 < 0 \wedge v(\frac{8}{3}y_1 + 2) = 3u_1 + 2. \end{aligned}$$

- $I_0 = \{2\}, I_1 = \{1\}$. *Then the indices from the first case are simply reversed and*

$$\psi_{I_0,I_1} \equiv -4u_1 - 3 < 0 \wedge v(\frac{8}{3}y_1 + 2) = -u_1 - 1.$$

- $I_0 = \emptyset, I_1 = \{1, 2\}$. *Then* $\mu_1 = 1$ *and*

$$\begin{aligned}\psi_{I_0,I_1} &\equiv t_1 = t_2 \wedge v(g_2 - g_1) > t_1 \\ &\equiv 4u_1 + 3 = 0 \wedge v(\frac{8}{3}y_1 + 2) > 3u_1 + 2.\end{aligned}$$

*In the first case, we can choose* $u_1 = -1$ *so that the first conjunct of* $\psi_{I_0,I_1}$ *is satisfied, and* $y_1 = 1$ *so that the second conjunct is satisfied for this value of* $u_1$. *Then* $t_{\mu_1} = t_2$ *evaluates to* 0, *and we can construct a value for* $y_2$ *in the original formula by adding* 1 *or* 2 *(plus optionally any positive integer multiple of* 3*) to* $g_2$, *which evaluates to* $\frac{1}{3}$. *So the assignments* $(u_1 = -1, y_1 = 1, y_2 = \frac{4}{3})$, $(u_1 = -1, y_1 = 1, y_2 = \frac{7}{3})$ *and* $(u_1 = -1, y_1 = 1, y_2 = \frac{34}{3})$ *are all satisfying assignments for the original formula.*

*In the second case, we can choose* $u_1 = 0$ *so that the first conjunct is satisfied, and* $y_1 = 1$ *so that the second one is too. Then* $t_{\mu_1} = t_1$ *evaluates to* 2, *and we can construct a value for* $y_2$ *in the original formula by adding for example* 9 *or* 18 *(plus optionally any positive integer multiple of* 27*) to* $g_1$, *which evaluates to* 5. *So the assignments* $(u_1 = 0, y_1 = 1, y_2 = 14)$, $(u_1 = 0, y_1 = 1, y_2 = 23)$ *and* $(u_1 = 0, y_1 = 1, y_2 = 68)$ *are all satisfying assignments of the original formula.*

*In the third case, the subformula* $4u_1 + 3 = 0$ *has no integer solution for* $u_1$, *so no solution for the original formula can be constructed from this disjunct.*

## 3.5 Complexity of the Decision Problem

Consider the language

$$\text{DivSLP} = \{(X, Y) : X, Y \text{ arithmetic circuits}, X \mid Y\}. \tag{3.10}$$

Recall that the binary relation $\mid$ expresses that its left argument divides its right argument. The next result gives a complexity bound for the decision problem for existential $L_{LVF}$-sentences, using DivSLP as an oracle.

**Proposition 13.** *The decision problem for existential* $L_{LVF}$*-sentences over* $\mathbb{Q}_p$ *(where* $p$, *given in binary, is regarded as part of the input) has complexity in* $\mathbf{NP}^{\text{DivSLP}}$.

*Proof.* The quantifier elimination procedure in Section 3.4 rewrites a given existential $L_{LVF}$-sentence $\varphi$ to an equivalent disjunction $\bigvee_{i \in I} \varphi_i$ of sentences of Presburger arithmetic. We claim that there is a nondeterministic polynomial-time algorithm, using DivSLP as an oracle, whose set of possible outputs on input $\varphi$ is $\{\varphi_i : i \in I\}$.

We turn the quantifier elimination procedure into a nondeterministic algorithm by guessing the partition $(I_0, \ldots, I_q)$ that determines the formula $\psi_{I_0,\ldots,I_q}$ in each elimination step. We represent rational constants of $p$-adic sort as pairs of arithmetic circuits (one for the numerator and one for the denominator). The reason for this is that each time we eliminate a variable, the bit length of the rational constants of $p$-adic sort in the formula potentially doubles (since we divide through by a coefficient to obtain (3.8) and subtract pairs of the resulting terms in (3.9)). But using arithmetic circuits, the representation length remains polynomial and arithmetic operations on integers can be done in unit time. Moreover for each integer constant $a$ of $p$-adic sort we can guess a value $k$ for $v(a)$ and verify that $k$ is the largest power of $p$ that divides $a$ with two calls to the DivSLP oracle. For a rational constant $b$ of $p$-adic sort we can then guess a value $k'$ and verify that $v(b) = k'$ using the homomorphism property (3.1). Note that integer constants of value sort remain small, that is, of polynomial bit length. In particular, the integer constants in the (pure Presburger) output formula are all small.

The key observation underlying the polynomial running time of the nondeterministic elimination procedure is that each elimination step takes a $L_{LVF}$ formula in the form (3.7) and produces a formula in the form (3.9) that has one fewer atom of the form $v(f) = s$ together with a polynomial-size pure Presburger formula on the side. In particular, after eliminating all existentially quantified $\mathbb{Q}_p$-variables, the resulting formula in Presburger arithmetic is of polynomial size. □

Next we use some recent results of Allender, Balaji and Datta [ABD14] (building on [ABH01]) on the complexity of decision problems for arithmetic circuits to show that DivSLP lies in the Counting Hierarchy.

Recall that a *threshold circuit* is a Boolean circuit with unbounded fan-in AND, OR, and MAJORITY gates, together with unary NOT gates. A family of such circuits is said to be *uniform* if it is **DLOGTIME**-uniform (see [ABD14] for more details). Allender, Balaji and Datta [ABD14, Theorem 2] show that there is a family $\{C_n\}$ of uniform threshold circuits of constant depth such that inputs of $C_n$ are indexed by pairs $(p, j)$ with $p < n^2$ prime and $1 \leq j \leq \lfloor \log p \rfloor$ that compute the following function:

- **Input** Integers $X$ and $Y$, with $1 \leq X, Y \leq 2^n$, in Chinese Remainder Representation, that is, two sequences of values indexed by $(p, j)$ giving the $j$-th bit of $X \bmod p$ and $Y \bmod p$ for each prime $p < n^2$.
- **Output** The $n$ most significant bits of $Y/X$.

It is immediate that the variant of the above problem whose output is whether or not $X$ exactly divides $Y$ also has a family of uniform threshold circuits. We can use this circuit family to derive a complexity bound for the language DivSLP via the following result:

**Proposition 14.** *[ABD14, Proposition 1] Let $L \subseteq \{0,1\}^*$ be a language such that for some $k$, some polynomial-time function $f$, and some uniform family of constant-depth threshold circuits $\{C_n\}$, it holds that $x \in L$ if and only if*

$$C_{2^{|x|^k}}(f(x,1), f(x,2), \ldots, f(x, 2^{|x|^k}))$$

*accepts. Then $L \in \mathbf{CH}$.*

**Proposition 15.** *The language* DivSLP *is in the Counting Hierarchy.*

*Proof.* We apply Proposition 14 to the family of threshold circuits $\{C_n\}$ that determine divisibility of integers in Chinese Remainder Representation. The function $f$ takes as input a pair of integers $X$ and $Y$, represented as arithmetic circuits, and a pair of integers $(p, j)$. If $p$ is prime then $f$ outputs the $j$-th bit of $X$ modulo $p$ and the $j$-th bit of $Y$ modulo $p$. □

The following is the main result of this chapter:

**Theorem 16.** *The decision problem for existential $L_{LVF}$-sentences over $\mathbb{Q}_p$ (where $p$, given in binary, is regarded as part of the input) has complexity within the Counting Hierarchy.*

*Proof.* By Proposition 13, the decision problem for existential $L_{LVF}$-sentences has complexity in $\mathbf{NP}^{\mathrm{DivSLP}}$. By Proposition 15, DivSLP $\in \mathbf{CH}$. Since $\mathbf{NP} \subseteq \mathbf{PP}$ the result follows. □

The complexity bound in Theorem 16 could be improved from **CH** to **NP** if one were able to give a polynomial bound on the size of the integer constants generated during the quantifier elimination procedure. We discuss the prospects for obtaining such a bound in the conclusion (Chapter 7).

**Remark 17.** *Given an existential $L_{LVF}$-sentence $\varphi$ of the form (3.6), we have noted that we can eliminate the quantifiers over the p-adic variables $x_1, \ldots, x_n$, thus obtaining a disjunction of existential sentences of Presburger arithmetic, with each disjunct having size bounded by a fixed polynomial in $|\varphi|$. Now it follows from Theorem 1 that a satisfiable quantifier-free formula $\varphi'$ of Presburger arithmetic can be satisfied by a tuple of integers of size at most polynomial in $|\varphi'|$. Thus we can assume that in a satisfying assignment $u_i \mapsto a_i$ of the original $L_{LVF}$-formula $\varphi$, each integer $v_p(a_i)$ has size polynomial in $|\varphi|$.*

# Chapter 4

# Existential Presburger Arithmetic with Divisibility

## 4.1 Introduction

Presburger arithmetic with divisibility is the first-order theory over the integers with addition, order and divisibility. While the decision problem for this theory is undecidable even in the case of only one quantifier alternation, the purely existential fragment was shown to be decidable by Lipshitz [Lip76], and independently in the same year by Bel'tyukov [Bel76].

In this chapter we will establish a new, tight upper bound on the size of the smallest solution of a satisfiable quantifier-free formula of Presburger arithmetic with divisibility. From this bound on the solution size we immediately get a new complexity upper bound for the decision procedure of the existential fragment of Presburger arithmetic with divisibility. For complexity purposes, we will allow multiplication by constants (but not variables) in formulas, and we will assume that all integers are represented in binary.

The existential fragment of Presburger arithmetic with divisibility can naturally be viewed as an extension of the existential fragment of (pure) Presburger arithmetic. The latter is known to be **NP**-complete (see Corollary 2), and moreover any satisfiable quantifier-free formula of Presburger arithmetic always has some satisfying assignment of size at most polynomial in the size of the formula [BT76]. In contrast, the smallest solution of a quantifier-free formula of Presburger arithmetic with divisibility can be of exponential size. Consider, for example, the formula

$$\bigwedge_{i=1}^{m+1} x_i > 1 \wedge \bigwedge_{i=1}^{m} (x_i \mid x_{i+1} \wedge x_i + 1 \mid x_{i+1})\,. \tag{4.1}$$

This formula states that each variable $x_{i+1}$ is divisible by both $x_i$ and its successor. Since $x_i$ and $x_i + 1$ are coprime if $x_i > 1$, it follows that $x_{i+1}$ is an integer multiple of $x_i(x_i + 1)$ and hence $x_{i+1} > x_i^2$ for each $i$. So we have $x_{m+1} > 2^{2^m}$, that is, every satisfying assignment of (4.1) has bit length at least exponential in $m$.

The main result of this chapter is to establish a matching singly exponential upper bound on the size of the smallest satisfying assignment of a satisfiable quantifier-free formula of Presburger arithmetic with divisibility. The bound is achieved through subtle adaptations to Lipshitz's original decision procedure in [Lip76] for the existential fragment of Presburger arithmetic with divisibility. As a corollary, we get a **NEXPTIME** upper bound for the decision problem of this theory, following from the existence of a simple guess-and-check procedure on the smallest solution of a formula.

The key idea behind Lipshitz's original algorithm is to transform a given existential sentence $\varphi$ into an equivalent disjunction of sentences $\varphi_i$, each of which satisfies a certain *local-to-global* principle, namely that each $\varphi_i$ is satisfiable in the integers $\mathbb{Z}$ if and only if it is satisfiable in the $p$-adic numbers $\mathbb{Q}_p$ for each prime $p \in P$ for some finite set $P$. Then decidability over $\mathbb{Z}$ follows from decidability of the existential first-order linear theory of the valued field $\mathbb{Q}_p$, see Chapter 3.

Unfortunately, in Lipshitz's construction it seems that the best possible bound one can achieve on the size of the $\varphi_i$ formulas is doubly exponential in the size of the input formula $\varphi$. This implies that the best upper complexity bound for the decision problem of existential Presburger arithmetic with divisibility that follows from Lipshitz's proof is **2NEXPTIME**. Our first technical contribution of this chapter, in Section 4.3, is to reformulate the transformation so that each formula $\varphi_i$ has size only singly exponential in that of $\varphi$. Roughly speaking, the idea is to replace sets of terms in each $\varphi_i$ by bases of the $\mathbb{Z}$-modules that they generate. Of course it must be verified that these formulas $\varphi_i$ still satisfy the local-to-global principle, which we do in Section 4.4.

## 4.2 Preliminaries

### 4.2.1 A Generalised Chinese Remainder Theorem

Recall the definition of the well-known Chinese remainder theorem from number theory:

**Theorem 18** (Chinese remainder theorem)**.** *Let $r_i \in \mathbb{Z}$, $m_i \in \mathbb{N}^+$ for $i = 1, \ldots, k$, with all the $m_i$ pairwise coprime. The $r_i$ are the* remainders*, and the $m_i$ are the*

moduli *or* divisors. *Then the system of congruences*

$$\begin{array}{rcll} x & \equiv & r_1 & (\text{mod } m_1) \\ x & \equiv & r_2 & (\text{mod } m_2) \\ & \vdots & & \\ x & \equiv & r_k & (\text{mod } m_k) \end{array} \tag{4.2}$$

*has a unique solution for $x$ modulo $\Pi_{i=1}^{k} m_i$.*

For the proofs in Section 4.4, we will need the following generalised version of this theorem.

**Theorem 19** (Generalised Chinese remainder theorem)**.** *Let $a_i, r_i \in \mathbb{Z}$, $m_i \in \mathbb{N}^+$ for $i = 1, \ldots, k$. Then the system of congruences*

$$\begin{array}{rcll} a_1 x & \equiv & r_1 & (\text{mod } m_1) \\ a_2 x & \equiv & r_2 & (\text{mod } m_2) \\ & \vdots & & \\ a_k x & \equiv & r_k & (\text{mod } m_k) \end{array} \tag{4.3}$$

*has a solution for $x$ if and only if $(a_i m_j, a_j m_i) | a_i r_j - a_j r_i$ and $(a_i, m_i) \mid r_i$ for all $i, j$. This solution is unique modulo* $\text{lcm}(m'_1, \ldots, m'_n)$*, where $m'_i = \frac{m_i}{(a_i, m_i)}$.*

Recall that we write $(a, b)$ for the greatest common divisor of $a$ and $b$. The proof of Theorem 19 can be found in [Mah58].

### 4.2.2 Integer Modules

A *module* over a ring is a generalisation of a vector space over a field. It is defined to be an abelian group under addition together with a multiplication operation between elements of the ring and the module, such that multiplication distributes over addition.

We consider submodules of $\mathbb{Z}^n$ over the ring of integers. A set of *generating vectors* for such a submodule $M \subseteq \mathbb{Z}^n$ is a set $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\} \subseteq M$ such that for all $\boldsymbol{v} \in M$, $\boldsymbol{v} = a_1 \boldsymbol{v}_1 + \ldots + a_k \boldsymbol{v}_k$ for some integer values of the $a_i$. A set of generating vectors of $M$ is a *basis* for $M$ if all the $\boldsymbol{v}_i$ are linearly independent, and in that case the coefficients $a_i$ are unique in the representation of each vector $\boldsymbol{v}$ as a linear combination of vectors from the basis. Every module $M \subseteq \mathbb{Z}^n$ has a basis, and every basis of $M$ has the same cardinality, which we call the *dimension* of $M$.

Let $M, N \subseteq \mathbb{Z}^n$ be submodules of $\mathbb{Z}^n$, each represented by a set of generating vectors. There are polynomial-time (in terms of the representation) algorithms for testing equality of $M$ and $N$ [Coh93, Section 2.4.3] and computing a basis of $M \cap N$ [Coh93,

Chapter 4, Exercise 18]. These algorithms work by reduction to the computation of Hermite normal forms of integer matrices. Here we will only need the following size bounds, which can be obtained from bounds on the size of the entries of matrices $U$ and $AU$ such that $AU$ is the Hermite normal form of a given integer matrix $A$.

**Theorem 20.** *Let $M, N \subseteq \mathbb{Z}^n$ be submodules of $\mathbb{Z}^n$, each represented by a set of generating vectors of size at most $s$. Then:*

*(i) If $M \neq N$ then there exists a vector in their symmetric difference of size at most polynomial in $s$.*

*(ii) The $\mathbb{Z}$-module $M \cap N \subseteq \mathbb{Z}^n$ has a basis of size at most polynomial in $s$.*

From now on, we will simply refer to submodules of $\mathbb{Z}^n$ as $\mathbb{Z}$-*modules*.

## 4.3 Syntactic Transformations of ∃PAD Formulas

### 4.3.1 Syntax and Conventions

Recall the definition of Presburger arithmetic (PA) from Section 2.2. *Presburger arithmetic with divisibility* (PAD) extends PA with a binary relation symbol $|$. Formally, PAD is defined to be the first-order logic over the structure $\langle \mathbb{Z}, +, \leq, |, 0, 1 \rangle$. For complexity purposes we also include a constant symbol $a$ for each $a \in \mathbb{Z}$, and we allow multiplication of a variable by a constant. We represent all integer constants in binary.

Terms of PAD are linear polynomials with integer coefficients. We write $f(\boldsymbol{x})$, $g(\boldsymbol{x})$, etc., for terms in integer variables $\boldsymbol{x} = x_1, \ldots, x_n$. Atomic formulas thus have the form $f(\boldsymbol{x}) = g(\boldsymbol{x})$, $f(\boldsymbol{x}) \leq g(\boldsymbol{x})$ or $f(\boldsymbol{x}) \mid g(\boldsymbol{x})$.

We are interested in deciding the truth of *existential* PAD-sentences (∃PAD-sentences) over the integers. It is not difficult to see that this problem reduces in nondeterministic polynomial time to the special case of sentences $\exists \boldsymbol{x}\, \varphi$ with $\varphi$ a conjunction of atomic formulas, i.e.,

$$\varphi \equiv A\boldsymbol{x} = \boldsymbol{b} \wedge C\boldsymbol{x} \geq \boldsymbol{d} \wedge \bigwedge_{i=1}^{m} f_i(\boldsymbol{x}) \mid g_i(\boldsymbol{x}) \tag{4.4}$$

for integer matrices $A$ and $C$ and integer vectors $\boldsymbol{b}$ and $\boldsymbol{d}$. This reduction is performed by rewriting the formula in negation normal form, i.e., pushing negations inwards

so that they are only applied to atomic formulas, replacing negated atoms using the equivalences

$$\begin{aligned}
\neg(f \le g) &\Leftrightarrow g+1 \le f\,,\\
\neg(f = g) &\Leftrightarrow f+1 \le g \vee g+1 \le f\,,\\
\neg(f \mid g) &\Leftrightarrow (f = 0 \wedge \neg(g=0)) \vee\\
&\qquad \exists x \exists y \big((g = x+y) \wedge (f \mid x)\\
&\qquad \wedge ((1 \le y \le f-1) \vee (1 \le y \le -f-1))\big)\,,
\end{aligned}$$

and finally using the distributive law to move all disjunctions to the outer level. In the third equivalence, the first disjunct covers the special case where $f$ evaluates to 0.[1], and the second disjunct covers the cases that $f \ge 2$ (in which case $1 \le y \le f-1$ must hold) or $f \le -2$ (in which case $1 \le y \le -f-1$ must hold). Clearly if $f = 1$ or $f = -1$ then the formula $\neg(f \mid g)$ can never evaluate to true.

The main result of this section is Theorem 23. This result corresponds to [Lip76, Lemma 4] and its proof uses the same basic ideas as the proof of that lemma. The main difference is that we introduce a semantic notion of *increasing formulas* in terms of $\mathbb{Z}$-modules. This reformulation is key to the singly exponential size bound on the formulas $\varphi_j$ in Theorem 23. No corresponding size bound is stated in [Lip76, Lemma 4]; in Section 4.4 we explain why the latter construction leads to an exponentially larger bound (see also [BI05, Section 5] for a similar accounting of the complexity of Lipshitz's decision procedure).

### 4.3.2 Eliminating Equalities and Inequalities

Let $\varphi(\boldsymbol{x})$ be a formula with free variables $\boldsymbol{x}$ and let $E$ and $\boldsymbol{u}$ be respectively a matrix and column vector of integers. We say that $\tilde{\varphi}(\boldsymbol{y})$ arises from $\varphi(\boldsymbol{x})$ by an *affine change of variables* $\boldsymbol{x} = E\boldsymbol{y} + \boldsymbol{u}$ if $\tilde{\varphi}(\boldsymbol{y})$ is obtained by substituting $E\boldsymbol{y} + \boldsymbol{u}$ for all free occurrences of $\boldsymbol{x}$ in $\varphi$.

The following construction will be used at several points in the sequel. Consider formula (4.4) and, applying Theorem 1 to the equalities and inequalities, let integer matrices $E^{(j)}$ and integer vectors $\boldsymbol{u}^{(j)}$, $j \in J$, be such that

$$\begin{aligned}
&\{\boldsymbol{x} \in \mathbb{Z}^n : A\boldsymbol{x} = \boldsymbol{b} \wedge C\boldsymbol{x} \ge \boldsymbol{d}\}\\
&= \bigcup_{j \in J} \{E^{(j)}\boldsymbol{y} + \boldsymbol{u}^{(j)} : \boldsymbol{y} \in \mathbb{Z}^{n-r}, \boldsymbol{y} \ge 0\}\,,
\end{aligned}$$

[1] By our definition of divisibility, $a \mid b$ if and only if there is an integer $c$ such that $ca = b$, so in particular, $0 \mid 0$ evaluates to true, and $0 \mid a$ evaluates to false for all $a \ne 0$.

where $r$ is the rank of $A$. For each $j \in J$ write

$$\varphi_j \equiv \boldsymbol{y} \geq \mathbf{0} \wedge \bigwedge_{i=1}^{m} \tilde{f}_{i,j}(\boldsymbol{y}) \mid \tilde{g}_{i,j}(\boldsymbol{y}), \tag{4.5}$$

where $\tilde{f}_{i,j}(\boldsymbol{y})$ and $\tilde{g}_{i,j}(\boldsymbol{y})$ arise from $f_i(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ by an affine change of variables $\boldsymbol{x} = E^{(j)}\boldsymbol{y} + \boldsymbol{u}^{(j)}$. From Theorem 1 we have:

**Proposition 21.** *Formulas $\varphi$ and $\bigvee_{j \in J} \varphi_j$ are equisatisfiable over the integers and each formula $\varphi_j$ has size at most $|\varphi|^{O(1)}$.*

### 4.3.3 Increasing Formulas and the Elimination Property

Consider a formula $\varphi \equiv \boldsymbol{x} \geq \mathbf{0} \wedge \bigwedge_{i=1}^{m} f_i(\boldsymbol{x}) \mid g_i(\boldsymbol{x})$, where $\boldsymbol{x} = x_1, \ldots, x_n$. Assume without loss of generality that each divisibility $f_i \mid g_i$ in $\varphi$ is *reduced* in the sense that the greatest common divisor of all coefficients of $f_i$ and $g_i$ is one.

We write $\mathbb{Z}[x_1, \ldots, x_k]$ for the set of all linear polynomials in variables $x_1, \ldots, x_k$ with integer coefficients, for $1 \leq k \leq n$. A *primitive term* is one such that the greatest common divisor of its coefficients is one. For any primitive term $f$ such that $af$ occurs on the left-hand side of a divisibility in $\varphi$ for some $a \in \mathbb{Z}$, define $M_f(\varphi) \subseteq \mathbb{Z}[x_1, \ldots, x_n]$ to be the smallest set such that

(i) $f \in M_f(\varphi)$,

(ii) $M_f(\varphi)$ is a $\mathbb{Z}$-module, that is, $M_f(\varphi)$ is closed under integer linear combinations (observing that linear integer polynomials are syntactically equivalent to integer vectors), and

(iii) for all terms $g$ and $h$ and for all $b \in \mathbb{Z}$, if $g \mid h$ is a divisibility in $\varphi$ and $bg \in M_f(\varphi)$, then $bh \in M_f(\varphi)$.

The following proposition shows a connection between the modules $M_f(\varphi)$ and satisfying assignments of the formula.

**Proposition 22.** *Suppose that $g \in M_f(\varphi)$ for some term $g$ and primitive term $f$ such that $af$ occurs on the left-hand side of a divisibility in $\varphi$. Then for every assignment $\boldsymbol{a} \in \mathbb{Z}^n$ that satisfies $\varphi$, $f(\boldsymbol{a})$ divides $g(\boldsymbol{a})$.*

*Proof.* The proof is by an easy case analysis over the definition of $M_f(\varphi)$. Take an arbitrary assignment $\boldsymbol{a} \in \mathbb{Z}^n$ that satisfies $\varphi$.

If $g \in M_f(\varphi)$ by rule (i), then $g \equiv f$ and the result trivially holds.

If $g \in M_f(\varphi)$ by rule (ii), then there exist terms $g_1, \ldots, g_m \in M_f(\varphi)$ and integers $b_1, \ldots, b_m$ such that $g = b_1 g_1 + \ldots + b_m g_m$, and by the inductive hypothesis, $g_i(\boldsymbol{a})$ is an integer multiple of $f(\boldsymbol{a})$ for all $i$. Since all the $b_i$ are integers, $g(\boldsymbol{a})$ is an integer multiple of $f(\boldsymbol{a})$.

Finally, if $g \in M_f(\varphi)$ by rule (iii), then there exists a term $g' \in M_f(\varphi)$ such that $\frac{1}{b}g' \mid \frac{1}{b}g$ is a subformula of $\varphi$ for some $b \in \mathbb{Z}$. Since $\boldsymbol{a}$ is a satisfying assignment of $\varphi$, $\frac{1}{b}g'(\boldsymbol{a})$ divides $\frac{1}{b}g(\boldsymbol{a})$, and multiplying both sides by $b$, we see that $g'(\boldsymbol{a})$ divides $g(\boldsymbol{a})$. By the induction hypothesis, $f(\boldsymbol{a})$ divides $g'(\boldsymbol{a})$, so it follows that $f(\boldsymbol{a})$ also divides $g(\boldsymbol{a})$ as required. □

Assume a total ordering $\chi$ on the variables appearing in $\varphi$, say $\chi \equiv 0 \leq x_1 \leq \ldots \leq x_n$. Let $\textsc{lv}(f)$ denote the leading variable of a term $f$, that is, the biggest variable with respect to $\chi$ which occurs with nonzero coefficient in $f$. We say that $\varphi$ is *increasing* with respect to this ordering if for each primitive term $f$ with leading variable $x_k$ such that $af$ occurs on the left-hand side of some divisibility in $\varphi$, $M_f(\varphi) \cap \mathbb{Z}[x_1, \ldots, x_k] = \mathbb{Z}f$, where $\mathbb{Z}f$ denotes the set of all integer multiples of $f$. That is, every linear term in $M_f(\varphi)$ is either an integer multiple of $f$ or includes a variable greater than $x_k$.

**Theorem 23.** *Let $\varphi \equiv \boldsymbol{x} \geq \boldsymbol{0} \wedge \bigwedge_{i=1}^{m} f_i(\boldsymbol{x}) \mid g_i(\boldsymbol{x})$ be a formula in variables $x_1, \ldots, x_n$. Then there is an equisatisfiable formula $\bigvee_{j \in J}(\chi_j \wedge \varphi_j)$, where $\chi_j$ specifies a total order on the variables appearing in $\varphi_j$ with respect to which $\varphi_j$ is increasing. Moreover each formula $\varphi_j$ has size at most $|\varphi|^{O(n)}$.*

*Proof.* We describe the construction of $\bigvee_{j \in J}(\chi_j \wedge \varphi_j)$ from $\varphi$ in three steps. There are various case analyses in each step and each branch of the construction yields one of the disjuncts $\chi_j \wedge \varphi_j$.

**Step 1.** Say that a term $f \in \mathbb{Z}[x_1, \ldots, x_n]$ is *positive* if all of its coefficients are positive, and that a formula of the form of $\varphi$ is positive if all of the terms that appear on the left side of a divisibility in the formula are positive. The first step is to transform $\varphi$ to a disjunction of positive formulas by a change of variables. To this end, given a sign vector $\boldsymbol{\sigma} \in \{-1, 1\}^m$, write $\chi_{\boldsymbol{\sigma}}$ for the formula $\bigwedge_{i=1}^{m} \sigma_i f_i \geq 0$. It is clear that $\varphi$ is equivalent to $\bigvee_{\boldsymbol{\sigma} \in \{-1,1\}^m}(\chi_{\boldsymbol{\sigma}} \wedge \varphi)$.

Applying Proposition 21 to one of the formulas $\chi_{\boldsymbol{\sigma}} \wedge \varphi$, we obtain an equisatisfiable formula $\bigvee_{k \in K}(\boldsymbol{y} \geq 0 \wedge \varphi_k(\boldsymbol{y}))$ in which each disjunct $\varphi_k(\boldsymbol{y})$ is a conjunction of divisibilities that arises from $\varphi$ by a change of variables. By construction, each term $f(\boldsymbol{y})$ appearing on the left side of a divisibility in $\varphi_k(\boldsymbol{y})$ has constant sign on $\mathbb{N}^n$. Such a term must have either all negative or all positive coefficients. By flipping positive

and negative coefficients, we can assume without loss of generality that $f$ has all positive coefficients.

Next we separately rewrite each positive formula $\varphi_k$ into an equisatisfiable disjunction of increasing formulas. In fact we describe how to rewrite an arbitrary positive formula $\psi(\boldsymbol{x})$ into an equisatisfiable disjunction of increasing formulas.

**Step 2.** Case split over all possible linear orderings $\chi$ of the variables $\boldsymbol{x}$ in $\psi$. For each such ordering $\chi$, we can check in polynomial time whether $\psi$ is increasing with respect to $\chi$: for each primitive term $f$ such that $af$ appears as the left side of a divisibility in $\psi$ for some integer $a$, calculate a basis for the module $M_f(\psi) \cap \mathbb{Z}[x_1, \ldots, \textsc{lv}(f)]$ and check if it is equal to $\{f\}$ or $\{-f\}$.

If $\psi$ is increasing with respect to $\chi$ then return $\chi \wedge \psi$. Otherwise proceed to Step 3.

**Step 3.** If $\psi$ is not increasing with respect to $\chi$ then there is a primitive term $f$ and a term $g \in M_f(\psi)$ such that $g$ is not an integer multiple of $f$ and $\textsc{lv}(g) \leq \textsc{lv}(f)$. Writing $S$ for the sum of the absolute values of the coefficients appearing in $g$, since $f$ is positive and includes a variable which is greater than or equal to all variables appearing in $g$, we know that $Sf + 1 > g$ and $-Sf - 1 < g$. Thus $f \mid g$ is equivalent to $\bigvee_{c=-S}^{S} cf = g$. Case splitting on $c$, pick a particular equality $cf = g$. Note that this equality is non-trivial by the assumption that $g$ is not an integer multiple of $f$.

By Proposition 21 we can replace $\psi \wedge \chi \wedge (cf = g)$ by an equisatisfiable disjunction $\bigvee_{l \in L}(\boldsymbol{y} \geq 0 \wedge \psi_l(\boldsymbol{y}))$, with each $\psi_l$ a conjunction of divisibilities and with vector $\boldsymbol{y}$ comprising one fewer variable than $\boldsymbol{x}$. We now proceed by case analysis on the formulas $\psi_l$ and return to Step 2. Note that since a substitution instance of a positive term under a map $\boldsymbol{x} = E\boldsymbol{y} + \boldsymbol{v}$ from $\mathbb{N}^n$ to $\mathbb{N}^{n-1}$ remains positive, we can assume that all terms on the left side of a divisibility in $\psi_l$ are positive.

This concludes the description of the procedure. It remains to bound the size of the resulting formulas. To this end, note that the only transformations performed on formulas are substitutions of terms for variables using Proposition 21. Each application of Proposition 21 causes a polynomial blow-up in the bit size of the integers in each formula. Moreover the number of times that we apply Proposition 21 along each branch of the above transformation (where the branch is determined by the resolution of each case analysis) is at most one plus the number $n$ of variables of the original formula.

The remaining potential source of a size blow-up is in Step 3: the case that $\psi$ is not increasing. Here the term $g$ lies in $M_f(\psi) \cap \mathbb{Z}[x_1, \ldots, x_k]$ but not $\mathbb{Z}f$, where $x_k$ is the leading variable of $f$. But by Theorem 20 there is such a function $g$ of size at most $|\psi|^{O(1)}$.

We conclude that the size of the constants in the output formula $\bigvee_{j\in J} \chi_j \wedge \varphi_j$ is at most $|\varphi|^{O(n)}$, being bounded by the composition of $O(n)$ polynomials of absolutely bounded degree. Note also that the number of conjuncts in each $\varphi_j$ is at most the number of conjuncts in $\varphi$. □

Consider a formula $\varphi$ that is increasing with respect to the variable ordering $0 \leq x_1 \leq \ldots \leq x_n$. We say that $\varphi$ has the *elimination property* if for each primitive term $f$ such that $af$ appears as the left side of some divisibility in $\varphi$ for some integer $a$, and for each $k$, the module $M_f(\varphi) \cap \mathbb{Z}[x_1, \ldots, x_k]$ is generated by terms $g_1, \ldots, g_s \in \mathbb{Z}[x_1, \ldots, x_k]$ such that the divisibilities $f \mid g_1, \ldots, f \mid g_s$ appear in $\varphi$. The name 'elimination property' follows terminology from [CLO07, Chapter 3].

Given an increasing formula, we construct an equivalent formula with the elimination property as follows. For each primitive term $f$, choose a basis $g_1, \ldots, g_s$ of $M_f(\varphi) \cap \mathbb{Z}[x_1, \ldots, x_k]$ and add divisibilities $f \mid g_1, \ldots, f \mid g_s$ to $\varphi$. Since the additional divisibilities do not change $M_f(\varphi)$, the resulting formula remains increasing.

## 4.4 Constructing Global Solutions

### 4.4.1 The Set of S-Terms

Consider a formula $\varphi(\boldsymbol{x}) \equiv \bigwedge_{i=1}^{m} f_i(\boldsymbol{x}) \mid g_i(\boldsymbol{x})$ in variables $\boldsymbol{x} = x_1, \ldots, x_n$ and a variable ordering $\chi(\boldsymbol{x}) \equiv 0 \leq x_1 \leq \ldots \leq x_n$. Let $f = a_0 + a_1x_1 + \ldots + a_kx_k$ and $g = b_0 + b_1x_1 + \ldots + b_kx_k$ be linear polynomials in $\varphi$ with $a_k, b_k \neq 0$, for some $1 \leq k \leq n$. Then the *S-polynomial* $S(f, g) = a_kg - b_kf$ of $f$ and $g$ is obtained by cancelling the leading variable in $f$ and $g$.[2]

Let $\mathit{Terms}(\varphi)$ denote the set of terms in $\varphi$. Consider the set of $S$-polynomials $\{S(f, g) : f, g \in \mathit{Terms}(\varphi), \mathrm{LV}(f) = \mathrm{LV}(g)\}$. This set potentially has cardinality quadratic in that of $\mathit{Terms}(\varphi)$. Extrapolating, the smallest set of terms that contains $\mathit{Terms}(\varphi)$ and is closed under taking $S$-polynomials has cardinality potentially doubly exponential in $n$.[3]

Lipshitz's original decidability proof [Lip76] involves closing the set of terms occurring in a given formula under the operation of forming $S$-polynomials, as described above. In Section 4.3 we have avoided such a construction, essentially by exploiting

[2]The name S-polynomial is short for *syzygy-polynomial*. The terminology is taken from work on Gröbner bases, see [CLO07, Chapter 2].

[3]Analysing the size of this set in specific cases seems quite hard. However there is a formal similarity between forming S-polynomials and performing elementary row operations in matrices. Using this connection one can translate an example from [FH97] to show that closing up under the formation of S-polynomials can lead to coefficients of magnitude doubly exponential in $n$.

the fact that all $S$-polynomials generated from $\mathit{Terms}(\varphi)$ lie in the $\mathbb{Z}$-module spanned by $\mathit{Terms}(\varphi)$. While the remaining part of the decidability proof is formally very similar to [Lip76], because we have earlier established the elimination property we are able to work with a restricted type of closure under forming $S$-polynomials, which involves only a singly exponential blow-up in the number of terms. The outcome is that we obtain a singly exponential bound on the size of the smallest satisfying valuation for a given formula as opposed to a doubly exponential bound from the proof of [Lip76].

To this end, we define the set $\mathit{STerms}(\varphi)$ of *S-terms associated with* $\varphi$ to be the smallest set that includes $\mathit{Terms}(\varphi)$ and if $f \in \mathit{Terms}(\varphi)$ and $g \in \mathit{STerms}(\varphi)$ then $S(f, g) \in \mathit{STerms}(\varphi)$. The cardinality of $\mathit{STerms}(\varphi)$ is bounded by $|\varphi|^{O(n)}$, and the size of the coefficients (represented in binary) in $\mathit{STerms}(\varphi)$ is bounded by $|\varphi|^{O(1)}$.

### 4.4.2 Combining $p$-adic Solutions

In this section, we will reduce the satisfiability problem for increasing conjunctions of divisibilities with the elimination property to the decision problem of the existential linear theory of the $p$-adic numbers. We showed in Chapter 3 that the complexity of the latter problem is in the Counting Hierarchy, and thus in **PSPACE**. The most important definition to recall from Chapter 3 is that of the $p$-adic valuation of a rational number $a$, for a given prime $p$:

$$v_p(a) = \begin{cases} \max\{k \in \mathbb{N} : p^k \mid a\} & \text{if } a \in \mathbb{Z} \setminus \{0\} \\ \infty & \text{if } a = 0 \\ v_p(b) - v_p(c) & \text{if } a = \frac{b}{c}\,. \end{cases}$$

Recall also that the set $\mathbb{Z}_p$ of $p$-adic integers is the set of $p$-adic numbers whose $p$-adic valuation is greater than or equal to zero.

Let $\varphi(\boldsymbol{x}) = \bigwedge_{i=1}^{m} f_i(\boldsymbol{x}) \mid g_i(\boldsymbol{x})$ be an increasing formula with respect to an ordering $\chi(\boldsymbol{x}) = 0 \leq x_1 \leq \ldots \leq x_n$. Suppose also that $\varphi$ satisfies the elimination property, as defined in Section 4.3. In this section we will abbreviate $M_f(\varphi)$ to $M_f$ for convenience.

Let $r = |\mathit{STerms}(\varphi)|$. Let $P_0$ be the set of primes $p$ such that either

- $p \leq m + r$, or
- $p$ divides a coefficient of some S-term, or
- there exist a primitive term $f$ and S-term $g$ such that $\mathbb{Z}g \cap M_f \neq \{0\}$ and $p$ divides the smallest positive integer $\lambda$ with $\lambda g \in M_f$, where $\{0\}$ is the $\mathbb{Z}$-module

that has the zero polynomial as its only element.[4]

We think of $P_0$ as the set of 'level-0 primes'. Note that the cardinality of $P_0$ is bounded by $|\varphi|^{O(n)}$ and each $p \in P_0$ has bit length at most $|\varphi|^{O(1)}$.

For a given prime $p$, a *p-adic integer solution* of $\varphi$ is a tuple $\boldsymbol{b} \in (\mathbb{Z}_p)^n$ such that $v_p(f(\boldsymbol{b})) \leq v_p(g(\boldsymbol{b}))$ for every divisibility $f \mid g$ occurring in $\varphi$. We know from Theorem 8 in Chapter 3 that if there is an integer vector $\boldsymbol{b} \in \mathbb{Z}^n$ such $\boldsymbol{b}$ is a $p$-adic integer solution of $\varphi$ for every prime $p$, then $b$ is also an integer solution of $\varphi$. The following main result of this chapter shows that in fact something more general due to the increasing and elimination properties of $\varphi$: to show the existence of an integer solution it suffices to find $p$-adic integer solutions for the primes in $p \in P_0$, and these solutions do not have to be integer-valued or identical for all such primes.

**Theorem 24.** *Let $\varphi(\boldsymbol{x})$ be an increasing formula with respect to an ordering $\chi(\boldsymbol{x}) = 0 \leq x_1 \leq \ldots \leq x_n$ which also satisfies the elimination property, and let $P_0$ be defined as above. Suppose that there exists a p-adic integer solution $\boldsymbol{b}_p \in (\mathbb{Z}_p)^n$ of $\varphi$ for each $p \in P_0$ such that $f_i(\boldsymbol{b}), g_i(\boldsymbol{b}) \neq 0$ for $i = 1, \ldots, m$.[5] Then there is an integer solution $\boldsymbol{a} \in \mathbb{Z}^n$ of $\varphi \wedge \chi$.*

*Proof.* We show how to generate $\boldsymbol{a} \in \mathbb{Z}^n$ with the following properties:

(i) For each prime $p \in P_0$, if $\mu_p$ is the maximum of the $p$-adic valuations $v_p(f(\boldsymbol{b}_p))$ for $f \in \mathit{Terms}(\varphi)$, then $\boldsymbol{a} \equiv \boldsymbol{b}_p \pmod{p^{\mu_p+1}}$;

(ii) For each prime $p \notin P_0$ and divisibility $f \mid g$ appearing in $\varphi$, we have $v_p(f(\boldsymbol{a})) \leq v_p(g(\boldsymbol{a}))$;

(iii) If $g \in \mathit{Terms}(\varphi)$ and $h \in \mathit{STerms}(\varphi)$ are such that $S(g, h)$ is not identically zero and $p \mid g(\boldsymbol{a}), h(\boldsymbol{a})$ for some prime $p \notin P_0$, then there exists a primitive term $f$ such that $M_f \cap \mathbb{Z}g \neq \{0\}$, $M_f \cap \mathbb{Z}h \neq \{0\}$, and $v_p(f(\boldsymbol{a})) = v_p(g(\boldsymbol{a})) = v_p(h(\boldsymbol{a}))$.

(iv) $f(\boldsymbol{a}) \neq 0$ for any $f \in \mathit{STerms}(\varphi)$ that is not identically zero.

Note that (i) and (ii) together imply that $v_p(f(\boldsymbol{a})) \leq v_p(g(\boldsymbol{a}))$ for every divisibility $f \mid g$ in $\varphi$ and every prime $p$. From this it immediately follows that $\boldsymbol{a}$ satisfies $\varphi$ from Theorem 8. The remaining conditions are necessary to ensure that we can always find a solution if one exists.

[4] Observe that from Theorem 20 it follows that the smallest $\lambda \in \mathbb{Z}$ such that $\lambda g \in M_f$ is of size at most polynomial in the size of (the generating set of) $M_f$.

[5] To avoid $p$-adic valuations of $\infty$, it is easy to have a number of special cases where some of the $f_i$ or $g_i$ evaluate to zero. See Chapter 3 for details.

We choose values for the variables in increasing order, as specified by $\chi$. The induction hypothesis is that after we have chosen values for $a_1, \ldots, a_k$, conditions (i)–(iv) hold for $a_1, \ldots, a_k$ and all terms $f, g, h$ that mention only variables $x_1, \ldots, x_k$.

Let the list $f_1 \mid g_1, \ldots, f_s \mid g_s$ comprise the divisibilities in $\varphi$ whose right-hand sides have leading variable $x_{k+1}$ and (for notational convenience) a trivial divisibility $1 \mid g$ for each $g \in \mathit{STerms}(\varphi)$ with leading variable $x_{k+1}$. Note that since $\varphi$ is increasing, no variables greater than $x_k$ appear on the left-hand sides of these divisibilities.[6]

Let $\boldsymbol{a}' = a_1, \ldots, a_k$ denote the values that have already been chosen. We derive a value $a_{k+1}$ such that conditions (i)–(iv) are satisfied by solving a system of congruences and non-congruences modulo powers of primes from the set

$$P_k = \{p \text{ prime} : p \in P_0 \text{ or } p \mid f(\boldsymbol{a}') \text{ for some nonzero } f \in \mathit{STerms}(\varphi) \text{ with } \textsc{lv}(f) \leq x_k\}\,.$$

Note that by item (iv) in the induction hypothesis, $P_k$ is finite.

(1) Given $p \in P_0$, let $\mu_p$ again be the maximum of the $p$-adic valuations $v_p(f(\boldsymbol{b}_p))$ for $f \in \mathit{Terms}(\varphi)$ for the given $p$-adic solution $\boldsymbol{b}_p$ of $\varphi$. Then we choose $a_{k+1}$ to satisfy the congruence:

$$a_{k+1} \equiv (\boldsymbol{b}_p)_{k+1} \pmod{p^{\mu_p+1}}\,. \tag{4.6}$$

(2) Next we consider $p \in P_k \setminus P_0$ such that $p$ does not divide $f_i(\boldsymbol{a}')$ for any $i = 1, \ldots, s$. In this case we choose $a_{k+1}$ to satisfy the following system of non-congruences:

$$g_i(\boldsymbol{a}', a_{k+1}) \not\equiv 0 \pmod{p} \quad \text{for all } i = 1, \ldots, s\,. \tag{4.7}$$

Write $g_i(x_1, \ldots, x_{k+1}) = h_i(x_1, \ldots, x_k) + c_i x_{k+1}$ for $i = 1, \ldots, s$. It must be that $(c_i, p) = 1$ for $i = 1, \ldots, s$ since $p \notin P_0$, so the system (4.7) is equivalent to

$$a_{k+1} \not\equiv -c_i^{-1} h_i(\boldsymbol{a}') \pmod{p} \quad \text{for all } i = 1, \ldots, s\,. \tag{4.8}$$

(3) Finally we consider $p \in P_k \setminus P_0$ such that $p \mid f_i(\boldsymbol{a}')$ for some $i = 1, \ldots, s$. Without loss of generality suppose that $f_1, \ldots, f_t$, for some $1 \leq t \leq s$, are the terms $f_i$ such that $p \mid f_i(\boldsymbol{a}')$. By Part (iii) of the induction hypothesis we know that these terms all have the same $p$-adic valuation, so we can take $w_p > 0$ to be the greatest power of $p$ that divides these $f_i(\boldsymbol{a}')$. We claim that there exists some integer $a_{k+1}$ satisfying the following system of congruences and non-congruences:

$$g_i(\boldsymbol{a}', a_{k+1}) \equiv 0 \pmod{p^{w_p}} \quad \text{for all } i = 1, \ldots, t \tag{4.9}$$

$$g_i(\boldsymbol{a}', a_{k+1}) \not\equiv 0 \pmod{p^{w_p+1}} \quad \text{for all } i = 1, \ldots, s\,. \tag{4.10}$$

[6] We can ignore divisibilities of the form $f \mid cf$ since these are trivially satisfied.

The claim is now that a value $a_{k+1}$ satisfying the systems (4.6), (4.8), (4.9) and (4.10) for all $p \in P_k$ always exists, and that for any such $a_{k+1}$, the assignment $a_1, \ldots, a_{k+1}$ satisfies conditions (i)–(iv).

For each prime $p \in P$, the congruence (4.6) clearly has exactly one solution modulo $p^{\mu_p+1}$.

For each prime $p \in P_k \setminus P_0$ such that $p$ does not divide $f_i(\boldsymbol{a}')$ for $i = 1, \ldots, s$, there is only one value for $a_{k+1}$ modulo $p$ that fails to satisfy any given non-congruence in (4.8). Now $s$ is at most the sum of the number of divisibilities in $\varphi$ and number of elements of $STerms(\varphi)$. Since $p \notin P_0$, it follows that $p > s$. Thus for each such prime $p$, the system of non-congruences (4.8) has a solution. $p^{w_p}$, so we can safely choose such a solution.

Next, consider the system (4.9) of congruences for a prime $p \in P_k \setminus P_0$ such that $p \mid f_i(\boldsymbol{a}')$ for some $i = 1, \ldots, s$. Recalling that $g_i(x_1, \ldots, x_{k+1}) = h_i(x_1, \ldots, x_k) + c_i x_{k+1}$, the congruences in (4.9) can be rewritten as

$$c_i a_{k+1} \equiv -h_i(\boldsymbol{a}') \pmod{p^{w_p}} \quad \text{for all } i = 1, \ldots, t\,. \tag{4.11}$$

Since $S(g_i, g_j) = c_i g_j - c_j g_i = c_i h_j - c_j h_i$, and noting that $p$ does not divide $c_i$ or $c_j$ since $p \notin P_0$, by the generalised Chinese remainder theorem (Theorem 19), the system (4.11) has a solution if and only if $(c_i p^{w_p}, c_j p^{w_p}) \mid S(g_i, g_j)(\boldsymbol{a}')$ for all $1 \leq i, j \leq t$. Clearly $(c_i, c_j)$ must divide $S(g_i, g_j)(\boldsymbol{a}')$, so the condition becomes $p^{w_p} \mid S(g_i, g_j)(\boldsymbol{a}')$ for all $1 \leq i, j \leq t$. Now, by assumption, $p \mid f_i(\boldsymbol{a}')$ and $p \mid f_j(\boldsymbol{a}')$. By Part (iii) of the induction hypothesis there exists a primitive term $f$ such that both $\mathbb{Z}f_i \cap M_f \neq \{0\}$ and $\mathbb{Z}f_j \cap M_f \neq \{0\}$, and $v_p(f(\boldsymbol{a}')) = v_p(f_i(\boldsymbol{a}')) = v_p(f_j(\boldsymbol{a}')) = w_p$. Since $f_i \mid g_i$ and $f_j \mid g_j$ are divisibilities in $\varphi$, by the transitivity property of $M_f$, we also have $\mathbb{Z}g_i \cap M_f \neq \{0\}$ and $\mathbb{Z}g_j \cap M_f \neq \{0\}$. In particular, there exists $\lambda \in \mathbb{Z}$ such that both $\lambda g_i \in M_f$ and $\lambda g_j \in M_f$. It follows that $\lambda S(g_i, g_j) \in M_f$. From the assumption that $p \notin P_0$ we can assume without loss of generality that $\lambda' S(g_i, g_j) \in M_f$ for some $\lambda' \in \mathbb{Z}$ coprime with $p$. By the induction hypothesis, we know that $\boldsymbol{a}'$ is a satisfying assignment of $\varphi$ restricted to terms and modules that mention only variables $x_1, \ldots, x_k$. By Proposition 22, it follows that $f(\boldsymbol{a})$ divides $\lambda' S(g_i, g_j)(\boldsymbol{a})$, and since $p^{w_p} \mid f(\boldsymbol{a})$ and $p$ and $\lambda'$ are coprime, we have $p^{w_p} \mid S(g_i, g_j)(\boldsymbol{a}')$, as required.

Now solutions $b_{k+1}$ of the congruences in (4.9) are defined modulo $p^{w_p}$, and so there are at least $p$ different solutions modulo $p^{w_p+1}$. Using a similar reasoning to that for system (4.8), since $p > s$, at least one of these solutions must also satisfy (4.10), so we can simultaneously satisfy the congruences in (4.9) and non-congruences in (4.10).

We now simply apply the Chinese remainder theorem to combine the solutions to these systems with the solutions to (4.6) and (4.8) to obtain a unique common solution modulo product $q$ of all the moduli involved. If a value $a_{k+1} < q$ generated in this way is not greater than all the $a_1, \ldots, a_k$, one can simply add a multiple of $q$ so that the order $\chi$ is satisfied. Finally, to ensure that (iv) holds, another multiple of $q$ might need to be added so that no nonzero S-term with leading variable $x_{k+1}$ evaluates to zero. Once these conditions hold, we have found our new value $a_{k+1}$.

We have shown that there is a common solution $a_{k+1}$ to all the above systems of congruences and non-congruences which respects the ordering $\chi$ and satisfies (iv). It remains to show that the assignment $a_1, \ldots, a_{k+1}$ satisfies conditions (i)-(iii). (i) must hold since we know that $a_i \equiv (\boldsymbol{b}_p)_i \pmod{p^{\mu_p+1}}$ for $1 \leq i \leq k$ from the induction hypothesis, and the same follows directly from (4.6) for $i = k+1$.

For (ii), take a prime $p \notin P_0$, and a divisibility $f \mid g$ in $\varphi$ such that $\textsc{lv}(g) = x_{k+1}$. Then there is a primitive term $f'$ such that $f = bf'$ for some integer $b$ and $g \in M_{f'}$. By the elimination property there are divisibilities $f' \mid h_1, \ldots, f' \mid h_\ell$ in $\varphi$ such that $g = c_1 h_1 + \ldots + c_\ell h_\ell$ for some integers $c_i$ with $\textsc{lv}(h_i) \leq x_{k+1}$ for all $i$. Since $p$ does not divide any coefficient of an S-term, $v_p(f(\boldsymbol{a}', a_{k+1})) = v_p(f'(\boldsymbol{a}', a_{k+1}))$. If $p \nmid f'(\boldsymbol{a}', a_{k+1})$ then $v_p(f(\boldsymbol{a}', a_{k+_1})) = 0 \leq v_p(g(\boldsymbol{a}', a_{k+1}))$. If $v_p(f'(\boldsymbol{a}', a_{k+1})) = w$, say, for $w > 0$, then for each $h_i$ with $\textsc{lv}(h_i) \leq x_k$, we know $v_p(h_i(\boldsymbol{a}', a_{k+1})) \geq w$ from the induction hypothesis, and for each $h_i$ with $\textsc{lv}(h_i) = x_{k+1}$, the systems (4.9) and (4.10) ensure that $v_p(h_i(\boldsymbol{a}', a_{k+1})) = w$. Since $g$ is an integer linear combination of the $h_i$, we have $w \leq v_p(g(\boldsymbol{a}', a_{k+1}))$ as required.

Next we show that the choice of $a_{k+1}$ determined above is such that condition (iii) remains true. To this end, suppose first that $p \mid g_i(\boldsymbol{a}', a_{k+1})$ and $p \mid g_j(\boldsymbol{a}', a_{k+1})$ for some $i, j \in \{1, \ldots, s\}$, where at least one of $g_i, g_j$ lies in $\mathit{Terms}(\varphi)$ and $S(g_i, g_j)$ is not identically zero. We must show that $M_f \cap \mathbb{Z}g_i \neq \{0\}$ and $M_f \cap \mathbb{Z}g_j \neq \{0\}$ for some primitive term $f$, and that $v_p(f(\boldsymbol{a}', a_{k+1})) = v_p(g_i(\boldsymbol{a}', a_{k+1})) = v_p(g_j(\boldsymbol{a}', a_{k+1}))$. Since $p \mid S(g_i, g_j)(\boldsymbol{a}')$ we have $p \in P_k$; moreover since we imposed the non-congruences (4.7) it must be that $p \mid f_\ell(\boldsymbol{a}')$ for some $\ell \leq t$. In turn it follows from the congruences (4.9) that $p \mid g_\ell(\boldsymbol{a}')$, whence $p \mid S(g_i, g_\ell)(\boldsymbol{a}')$ and $p \mid S(g_j, g_\ell)(\boldsymbol{a}')$.

By condition (iii) in the induction hypothesis, there must be a primitive term $f$ with $v_p(f(\boldsymbol{a}')) = w$, say, such that $M_f$ has nonzero intersection with each of $\mathbb{Z}S(g_i, g_\ell)$, $\mathbb{Z}S(g_j, g_\ell)$ and $\mathbb{Z}f_\ell$, and with $v_p(S(g_i, g_\ell)(\boldsymbol{a}')) = v_p(S(g_j, g_\ell)(\boldsymbol{a}')) = v_p(f_\ell(\boldsymbol{a}')) = w$. Since $f_\ell \mid g_\ell$ occurs in $\varphi$, by the transitivity property of $M_f$ we also have that $M_f \cap \mathbb{Z}g_\ell \neq \{0\}$, and the systems (4.9) and (4.10) ensure that $v_p(g_\ell(\boldsymbol{a}', a_{k+1})) = w$. Since $S(g_i, g_\ell) = c_i g_\ell - c_\ell g_i$ with $p \nmid c_i$ and $p \nmid c_\ell$, it follows in turn that $M_f \cap \mathbb{Z}g_i \neq \{0\}$

and $v_p(g_i(\boldsymbol{a}', a_{k+1})) \geq w$, with the non-congruences (4.10) further ensuring that $v_p(g_i(\boldsymbol{a}', a_{k+1})) = w$. We can similarly show that $M_f \cap \mathbb{Z}g_j \neq \{0\}$ and $v_p(g_\ell(\boldsymbol{a}', a_{k+1})) = w$, completing the argument.

The second situation in which we must establish condition (iii) is when $p \mid g_i(\boldsymbol{a}', a_{k+1})$ and $p \mid f(\boldsymbol{a}')$ for some $f, g_i \in \mathit{STerms}(\varphi)$ such that $f$ has leading variable at most $x_k$ and at least one of $f$ or $g_i$ is in $\mathit{Terms}(\varphi)$. Since $p \mid f(\boldsymbol{a}')$ it must be that $p \in P_k$. Then, as in the previous case, we argue that $p \mid f_\ell(\boldsymbol{a}')$ for some $\ell \leq t$ and proceed similarly. □

For a bound on the size of the solution generated by this procedure, observe that the bit length of the $\mu_p$ is polynomial in $n$ and $m$ by Remark 17 in Chapter 3. We can assume without loss of generality that $m$ is bounded by a polynomial in $n$ (otherwise some divisibilities would be linearly dependant), so that the bit length of the $\mu_p$ is polynomial in $n$. In the first stage of the procedure, when we generate a solution for $x_1$, the system (4.6) is the only one we need to consider. The number of congruences in this system is bounded by $|\varphi|^{O(n)}$, and each modulus has bit length at most $|\varphi|^{n^{O(1)}}$ as $\mu_p$ has polynomial size in $n$. It follows that the product of all the moduli, and thus the generated value $a_1$ has size bounded by $|\varphi|^{n^{O(1)}}$. In the $(k+1)$-th stage of the procedure, we can assume that the bit length of all the computed values $a_1, \ldots, a_k$ is bounded by $|\varphi|^{n^{O(1)}}$. The primes that appear in the systems (4.7), (4.9) and (4.10) have size at most $|\varphi|^{n^{O(1)}}$, and the $p$-adic valuation $w_p$ has polynomial size in $n$, so again, we obtain an upper bound of $|\varphi|^{n^{O(1)}}$ on the size of the product $q$ of all the moduli, and hence an overall exponential upper bound on the size of the solution $\boldsymbol{a}$. Since the transformation of a formula of the form (4.4) to an increasing formula with the elimination property in Section 4.3 caused an exponential blow-up (in the number of variables $n$) on the size of the coefficients, we can conclude that given an arbitrary formula in existential Presburger arithmetic with divisibility, if the formula has a solution then it has one of exponential bit length in the number of variables. This gives us the following main result of this chapter.

**Theorem 25.** *Let $\varphi(x_1, \ldots, x_n)$ be an arbitrary ∃PAD formula. If $\varphi$ has an integer solution, then it has a solution $(a_1, \ldots, a_n) \in \mathbb{Z}^n$, where the bit length of each $a_i$ is bounded by $|\varphi|^{n^{O(1)}}$.*

**Corollary 26.** *The decision problem for ∃PAD formulas is in **NEXPTIME**.*

Note that our $|\varphi|^{n^{O(1)}}$ bound does not yet provide a concrete algorithm for solving the decision problem for ∃PAD in nondeterministic exponential time, but merely

proves the existence of such an algorithm. Additional calculations would be necessary to replace $O(1)$ in the bound with an actual constant, which would yield a guess-and-check procedure on the smallest solution of any given quantifier-free formula of PAD.

# Chapter 5

# Synthesis Problems for One-Counter Automata

## 5.1 Introduction

In this chapter, we will work with one-counter automata whose actions are additions of integers to the counter and whose guards are equality tests, and with a parameterised version of them that allows parameters on both actions and (implicitly) guards. We will simply refer to these models as one-counter automata and parameterised one-counter automata, respectively, in this chapter.[1] We will study two different synthesis problems for parameterised one-counter automata, a synthesis problem with universal co-Büchi acceptance condition and a more general LTL synthesis problem.

We will mostly follow the definitions and notation from [HKOW09] and [GHOW10]. In [HKOW09], reachability problems for both non-parameterised and parameterised one-counter automata were studied, and [GHOW10] presented new results on linear time and branching time model checking problems for parameterised one-counter automata. These model checking problems treat the parameters *universally*: one asks that a specification be satisfied for all parameter values. In other words, one asks for correctness in all contexts.

The equivalent synthesis problems, where parameters are treated *existentially*, were shown to be undecidable in the case of computation tree logic (CTL), and stated as an open problem in the case of linear time logic (LTL) in [GHOW10]. Intuitively, LTL synthesis generalises a variety of situations where an infinite-state system is only

[1] Recall that the next chapter will use a different definition of one-counter automata, which we will again simply refer to as one-counter automata there. We believe that no confusing will arise from this as the two definitions are local to their respective chapters.

partly given, and we wish to complete it to ensure that it is guaranteed to satisfy certain requirements.

We study this general LTL synthesis problem in this chapter, asking whether there exist parameter values in a given automaton such that all infinite computations from the initial configuration satisfy a given LTL specification. We show a reduction from this problem to the decision problem in a certain fragment of Presburger arithmetic with divisibility via the more specific universal co-Büchi synthesis problem and a reachability problem in parameterised one-counter automata. To this end, we also give a detailed analysis of the results from [HKOW09] concerning this reachability problem, which asks whether there exist values for the parameters such that a given target configuration can be reached from a given start configuration.

The basic synthesis problem we consider, which we call universal co-Büchi synthesis, asks whether there exist parameter values for a given parameterised one-counter automaton such that all infinite computations that start from the initial configuration visit each state from a given set of accepting states only finitely many times. We first express the negation of this problem in terms of reachability. This gives us a reduction from the negation of universal co-Büchi synthesis to the decision problem of a fragment of Presburger arithmetic with divisibility which was claimed to be decidable in [BI05]. This fragment allows one quantifier alternation in formulas but places restrictions on where variables may occur within divisibilities. A gap in the decidability proof in [BI05] was recently discovered, and the decidability result for this fragment of Presburger arithmetic with divisibility is currently under review.

The reason why we chose to encode the negation of the universal co-Büchi synthesis problem, rather than the problem in its positive form, is that the details of the calculations are somewhat simpler and easier to follow.

The translation of reachability to ∃PAD, and thus of universal co-Büchi synthesis to PAD with one quantifier alternation and restricted divisibilities, is based on a symbolic encoding of computations of the automaton, following the idea from [HKOW09] of representing any given computation by a constant number of flows in a network, which serve as polynomial-size certificates of reachability. Intuitively, in the negation of the universal co-Büchi synthesis problem (is it true that for all possible values of the parameters, there is at least one computation that visits each accepting state infinitely often), the integer-valued parameters in the automaton correspond to universally quantified variables in the resulting PAD formula with one quantifier alternation, and the existence of a computation with the desired property is encoded via existential quantifiers.

We then give a reduction from the synthesis problem for LTL specifications to the universal co-Büchi synthesis problem. The reduction follows the same general ideas as standard LTL model checking on Kripke structures.

Finally, we prove $\mathbf{NP}^{\mathbf{NP}}$-hardness of universal co-Büchi synthesis using a reduction from a generalised version of the subset sum problem. This reduction also gives an intuition of the kinds of properties that can be expressed in terms of this problem.

## 5.2 Preliminaries: One-Counter Automata

In general, a one-counter automaton is a finite automaton with an associated counter that can store a single value. Every transition can perform a read or write operation on the counter.

Different types of one-counter automata have been defined in the literature, and we consider two of these types in this thesis. In this chapter, we will refer as one-counter automata to automata whose counter ranges over the nonnegative integers, with the counter values and updates encoded in binary, and with each transition being an addition of an integer to the counter, or a zero test on the current counter value. In Chapter 6, the term one-counter automaton will take on a slightly different meaning. We will also use different definitions of configurations and computations in one-counter automata in both chapters, with the difference that we only allow valid configurations and computations in this chapter, while in Chapter 6 we relax the definition to allow invalid versions of them to simplify the following proofs. We believe that no confusion will arise from this, as each meaning is restricted to its respective chapter.

Formally, a *one-counter automaton* (1-CA) is a tuple $\mathcal{C} = (V, E, \lambda)$, where $V$ is a finite set of states, $E \subseteq V \times V$ is a set of transitions between states, and $\lambda : E \to Op$ labels each transition with an operation on the counter, where $Op = \{\text{zero?}\} \cup \{\text{add}(a) : a \in \mathbb{Z}\}$. Intuitively, a zero test (i.e., a transition labelled zero?) can only be taken if the current counter value is zero, and it leaves the counter value unchanged. An action (i.e., a transition labelled add($a$)) adds $a$ to the counter and can only be taken if the resulting value is nonnegative.

A *configuration* of a 1-CA $\mathcal{C}$ is a pair $(v, c)$, for some $v \in V$ and $c \in \mathbb{N}$, with the intuitive meaning that the automaton is in state $v$ with counter value $c$.

The transition relation $E$ between states induces an unlabelled transition relation between configurations: every transition $v \longrightarrow v'$ with label zero? induces a transition

$(v, 0) \longrightarrow (v', 0)$, and every transition $v \longrightarrow v'$ with label add($a$) induces a set of transitions $\{(v, c) \longrightarrow (v', c') \mid c, c' \geq 0,\ c' - c = a\}$. Note that 1-CA are nondeterministic, so there may be several outgoing transitions from a single configuration.

A *computation* $\pi$ of $\mathcal{C}$ is a (finite or infinite) sequence of transitions between configurations $\pi = (v_0, c_0) \longrightarrow (v_1, c_1) \longrightarrow (v_2, c_2) \longrightarrow \cdots$. If $\pi_1$ is a finite computation from $(v, c)$ to $(v', c')$ and $\pi_2$ is a computation starting in $(v', c')$, we write $\pi_1\pi_2$ for the concatenation of both computations.

Testing the current counter value for equality with an arbitrary natural number $a$ can easily be simulated in this model by a sequence of three transitions, as shown in Figure 5.1. In graphs, we simplify labels of transitions by writing $+a$ for add($a$), $-a$ for add($-a$), etc.

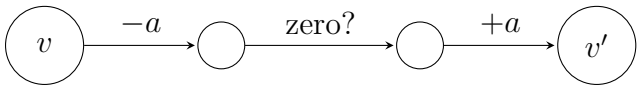


Figure 5.1: A gadget simulating an equality test with an arbitrary natural number $a$. From $(v, c)$, the three transitions can be taken if and only if $c = a$.

A *parameterised one-counter automaton* is a tuple $\mathcal{C} = (V, E, X, \lambda)$, where $V$ and $E$ are defined as for 1-CA, $X$ is a finite set of integer parameters, and $\lambda : E \to Op$ labels each transition with an operation from $Op = \{\text{zero?}\} \cup \{\text{add}(a),\ \text{add}(ax) : a \in \mathbb{Z},\ x \in X\}$. Substituting integer values for the parameters results in a 1-CA as defined above.

Note that every 1-CA $\mathcal{C}$ without zero tests is syntactically equivalent to a weighted graph (see Section 2.4.1 for definitions), where a transition labelled add($a$) is seen as having weight $a$. However, the two are semantically different, as 1-CA introduce the additional restriction of the counter being nonnegative. We write $G_\mathcal{C}$ for the weighted graph corresponding to $\mathcal{C}$ in that way. Similarly, any parameterised 1-CA $\mathcal{C}$ which does not have any zero tests is syntactically equivalent to a parameterised weighted graph $G_\mathcal{C}$. For a computation $\pi$ in a given (non-parameterised or parameterised) $\mathcal{C}$ and a path $\gamma$ in the corresponding weighted graph $G_\mathcal{C}$, we say that $\pi$ is a computation over $\gamma$ if both follow the same transitions.

For a given 1-CA $\mathcal{C} = (V, E, \lambda)$ with initial configuration $(v, c)$ and target configuration $(v', c')$, the *reachability* problem asks whether there is a computation from $(v, c)$ to $(v', c')$ in $\mathcal{C}$. The following result [LLT04, Lemma 42] shows that this problem is in PSPACE.

**Proposition 27.** *There is a polynomial $P$ such that, given a 1-CA $\mathcal{C}$ and configurations $(v, c)$ and $(v', c')$, if there is a computation from $(v, c)$ to $(v', c')$ then there is such a computation of length at most $2^{P(n)}$, where $n$ is the maximum of $|\mathcal{C}|$ and the bit lengths of $c$ and $c'$.*

Given a parameterised 1-CA $\mathcal{C} = (V, E, X, \lambda)$ and two configurations $(v, c)$ and $(v', c')$, the *reachability* problem asks if there are values for the parameters such that there is a computation from $(v, c)$ to $(v', c')$ in $\mathcal{C}$.

**Example 28.** *Intuitively, it is easy to see that various conditions in the form of equalities, inequalities and divisibilities on linear terms can be encoded in terms of reachability in parameterised 1-CA. Figure 5.2 shows a simple example of how a linear inequality can be encoded as a reachability problem. Figure 5.3 shows an example for a linear equality, and Figure 5.4 for a linear divisibility. We will discuss this connection between reachability in 1-CA and relations between linear terms in detail in the following sections.*

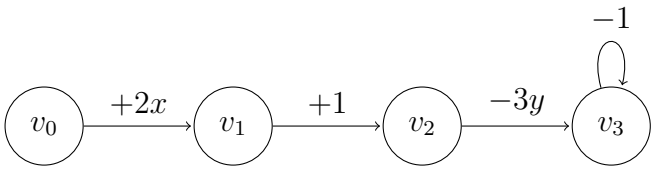


Figure 5.2: Reachability encoding an inequality: $(v_3, 0)$ is reachable from $(v_0, 0)$ if $x \geq 0$ and $3y \leq 2x + 1$.

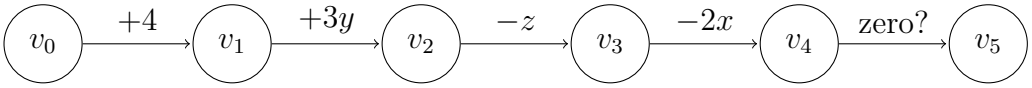


Figure 5.3: Reachability encoding an equality: $(v_5, 0)$ is reachable from $(v_0, 0)$ if $2x + z = 3y + 4$ (and also $3y + 4 \geq 0$, $3y + 4 \geq z$).

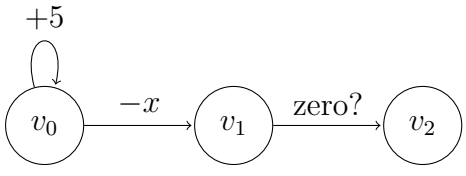


Figure 5.4: Reachability encoding divisibility: $(v_2, 0)$ is reachable from $(v_0, 0)$ if $x$ is a nonnegative integer multiple of 5.

The synthesis problem with universal co-Büchi acceptance condition, or *universal co-Büchi synthesis* problem for parameterised 1-CA asks, given a parameterised 1-CA $\mathcal{C} = (V, E, X, \lambda)$ with a set of accepting states $V_{Acc} \subseteq V$ and an initial configuration $(v, c)$, if there are parameter values for which on all infinite computations starting from $(v, c)$, all states in $V_{Acc}$ are visited only finitely often.

The *LTL synthesis* problem for parameterised 1-CA is to decide, for a given LTL formula $\varphi$ over the set of propositions $P$ and for a parameterised 1-CA $\mathcal{C} = (V, E, X, \lambda)$ with initial configuration $(v, c)$ and with transitions labelled with elements of $2^P$, whether there exist parameter values for which $\varphi$ holds on all infinite computations from $(v, c)$, i.e., it is satisfied by the sequence of labels along all such computations.

## 5.3 Overview

The reachability problem for 1-CA (without parameters) was shown to be **NP**-complete in [HKOW09]. Reductions in both directions between reachability for parameterised 1-CA and the decision problem for ∃PAD have been shown in the same paper, formalising the intuition given in Example 28 in the previous section. We first describe their reduction from ∃PAD to reachability. This reduction is quite simple; the other direction requires more effort and will be treated in detail in Section 5.4.

**Proposition 29** ([HKOW09])**.** *There is a nondeterministic polynomial-time reduction from the decision problem for ∃PAD to the reachability problem for parameterised 1-CA.*

*Proof.* Let $\varphi$ be a quantifier-free PAD formula. Without loss of generality, we can assume that $\varphi$ is a positive Boolean combination of atomic formulas of the form $f = g$, $f \leq g$ and $f \mid g$. (We eliminate negations in the same way as in Section 4.3.1). For each such atomic subformula $\psi$, we build a parameterised 1-CA with a distinguished start location $v_\psi$ and target location $v'_\psi$, with the parameters of the automaton corresponding to the variables in the input formula. In our construction, there are values for the parameters such that there exists a computation from $(v_\psi, 0)$ to $(v'_\psi, 0)$ if and only if $\psi$ is satisfied for the corresponding values of the variables in $\psi$. The parameters in the automaton correspond to the variables in the input formula. We then combine these automata into a 1-CA representing $\varphi$ using sequential composition to model conjunction and nondeterminism to model disjunction.

First we nondeterministically choose a sign for each variable and each term in $\varphi$. For example, in the subformula $\psi_1 \equiv x_1 + 2x_2 = -4x_3 + 2$, we might guess that

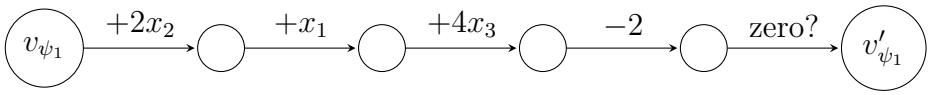


Figure 5.5: A subautomaton representing the formula $\psi_1 \equiv f = g$, where $f \equiv x_1 + 2x_2$ and $g \equiv -4x_3 + 2$. $(v_{\psi'_1}, 0)$ is reachable from $(v_{\psi_1}, 0)$ iff $\psi_1$ holds with the nondeterministically chosen signs.

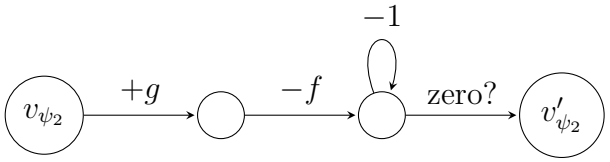


Figure 5.6: A subautomaton representing the formula $\psi_2 \equiv f \leq g$. For simplicity, the actions $+g$ and $-f$ are represented as only one transition each.

$x_2$ is nonnegative, $x_1$ and $x_3$ are less than or equal to zero, and both $x_1 + 2x_2$ and $-4x_3 + 2$ are nonnegative. Then we construct the automaton in Figure 5.5. If $x_1$ had been guessed to be nonnegative, we could have swapped the order of the first two transitions. If both the left and right side of the equality had been guessed to be zero or negative, the signs of the transitions would have been changed and the order of the changed to prevent the counter from going negative.

In the case of a subformula which is an inequality, i.e., $\psi_2 \equiv f \leq g$, if we guess that both $g$ and $f$ are nonnegative, the resulting subautomaton is shown in Figure 5.6.

Finally, for $\psi_3 \equiv f \mid g$, again assuming that both $f$ and $g$ are nonnegative, the corresponding subautomaton is as in Figure 5.7.

□

The outline given in [HKOW09] of how to translate an instance of reachability in the parameterised case to a ∃PAD formula omits some details and does not address the complexity of the procedure, so we will go through a more precise explanation including a detailed complexity assessment in the next section, making use of a symbolic encoding of the Bellman-Ford algorithm to obtain a reduction in nondeterministic polynomial

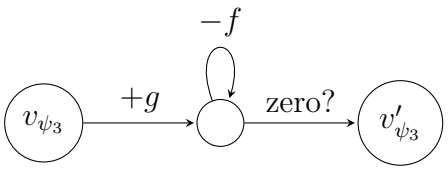


Figure 5.7: A subautomaton representing the formula $\psi_3 \equiv f \mid g$. Again, loading a linear term onto the counter is represented as only one transition for simplicity.

time. For a description of the Bellman-Ford algorithm, see Section 2.4.2.

Our analysis relies on a symbolic representation of computations without zero tests in terms of reachability certificates, developed in [HKOW09]. We first recall the definition and then give a symbolic encoding of these certificates in ∃PAD. We then use this result for an encoding of general computations in 1-CA.

As a result of the translation, we will derive the precise form of an ∃PAD formula obtained by the reduction from reachability, which can be used to express the negation of an instance of universal co-Büchi synthesis as a formula in PAD with one quantifier alternation, and with existentially quantified variables being restricted to only occur on the right-hand side of a divisibility atom. This fragment of PAD was claimed to be decidable in [BI05]. We will discuss reachability in Section 5.4, universal co-Büchi synthesis in Section 5.5.1, and LTL synthesis in Section 5.5.2.

## 5.4 Encoding Parameterised Reachability in existential PAD

### 5.4.1 General Description

Let $\mathcal{C}$ be a 1-CA, and let $\mathcal{C}'$ be the 1-CA that is identical to $\mathcal{C}$ except that it does not have any zero tests. Given a path $\gamma = e_1 e_2 \dots e_n$ in $G'_{\mathcal{C}}$ with $\text{start}(e_1) = v$ and $\text{end}(e_n) = v'$, we define $\text{drop}(\gamma) = \text{weight}(\gamma')$, where $\gamma'$ is the prefix of $\gamma$ with minimum weight. We can then say that $\gamma$ corresponds to a valid computation from $(v, c)$ to $(v', c')$ if and only if $\text{weight}(\gamma) = c' - c$ and $\text{drop}(\gamma) \geq -c$.

For a path $\gamma$ from $v$ to $v'$ in $G'_{\mathcal{C}}$, if there is a computation $\pi$ over $\gamma$ from $(v, c)$ to $(v', c')$ in $\mathcal{C}$, we say that the path $\gamma$ can be *taken from* the configuration $(v, c)$ and *taken to* the configuration $(v', c')$ in $\mathcal{C}$. Since computations in a 1-CA are more restricted than paths in a weighted graph in the sense that transitions can only be taken if the counter value remains nonnegative, it is possible that a path can be taken from (or to) $(v, c)$ but not $(v, c')$ for some state $v$ with $c' \neq c$.

**Example 30.** *In Figure 5.8, for any given value of $x$, the cycle $(v_1, v_2)(v_2, v_3)(v_3, v_1)$ can be taken from $(v_1, c)$ if and only if $c \geq -x + 20$, and taken to $(v_1, c')$ if and only if $c' \geq \max(5, x - 15)$.*

Let $f$ be a flow in the weighted graph $G'_{\mathcal{C}}$ (for definitions concerning flow networks, see Section 2.4.1). A cycle in $f$ is a cycle $\omega$ in $G'_{\mathcal{C}}$ such that every edge of $\omega$ lies in the support of $f$. A flow $f$ is a *reachability certificate* for two configurations $(v, c)$ and $(v', c')$ if there is a computation $\pi$ from $(v, c)$ to $(v', c')$ over a path $\gamma$ such that $f = f_\gamma$,

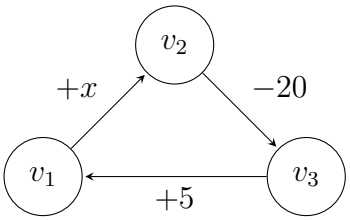


Figure 5.8

i.e., $f$ is a path flow corresponding to $\gamma$, and one of the following conditions holds: (i) $f$ has no positive cycles, (ii) $f$ has no negative cycles, (iii) there exists a positive cycle $\omega$ that can be taken from $(v, c)$ and a negative cycle $\omega'$ that can be taken to $(v', c')$. We will call the three cases type-1, type-2 and type-3 reachability certificates, respectively.

**Proposition 31** ([HKOW09])**.** *If $(v', c')$ is reachable from $(v, c)$ in a 1-CA $\mathcal{C}$ without zero tests, then there is a computation $\pi = \pi_1\pi_2\pi_3$ that starts in $(v, c)$ and ends in $(v', c')$, where $\pi_1$, $\pi_2$ and $\pi_3$ each yield a polynomial size (in $|\mathcal{C}|$ and the bit lengths of $c$ and $c'$) reachability certificate, of type 1, 3 and 2, respectively.*

Note that the definition of reachability certificates relies on the fact that $\mathcal{C}$ has no parameters. However, Proposition 31 can be applied to parameterised 1-CA in the sense that substituting integer values for the parameters yields a non-parameterised 1-CA to which the proposition applies.

The first step in the translation of reachability in parameterised 1-CA to the decision problem for ∃PAD is to encode the existence of each type of reachability certificate (from a given configuration $(v, c)$ to a given configuration $(v', c')$) as a ∃PAD formula. We will call these formulas $\varphi_{type1}^{(v,c),(v',c')}(\boldsymbol{x})$, $\varphi_{type2}^{(v,c),(v',c')}(\boldsymbol{x})$ and $\varphi_{type3}^{(v,c),(v',c')}(\boldsymbol{x})$, respectively, where the variables $\boldsymbol{x}$ of the formula correspond to the parameters of the automaton.

We represent flows as vectors of variables, with one variable for each transition in the automaton, representing the flow assigned to that particular transition. The main idea for all three translations is the same. For a reachability certificate $f$ from $(v, c)$ to $(v', c')$ we need to encode the following conditions.

- $f$ is a path flow from $v$ to $v'$. We can encode this using linear constraints corresponding to flow conservation properties and the fact that $|f|$ is equal to 1.
- $\text{weight}(f) = c' - c$. The main difficulty here is that a formula representing $\text{weight}(f)$ will generally contain products of the variables representing the flow $f$ and those representing the parameters of the automaton. One way to deal

with this problem is to make a change of variables and introduce divisibilities to express these products.

- $\mathrm{drop}(\gamma) \geq -c$ for some path $\gamma$ that corresponds to $f$. Note that there may be more than one such path, but we can nondeterministically choose the right one using the notion of decompositions of flows, which will be defined in the next section. We will again express products of variables as divisibilities.
- Finally, we need to express the existence or non-existence of positive or negative cycles. While it would be possible to use a disjunction or conjunction over all cycles in the graph, this could lead to an exponential blow-up of the size of the formula. To avoid this, we can use a symbolic encoding of the Bellman-Ford algorithm, which checks for negative cycles in a weighted graph in polynomial time. Since the reduction is nondeterministic, in the case of existence (but not non-existence) of positive or negative cycles, we can also simply use a guess-and-check procedure.

The details of each step of the encoding of reachability certificates will be explained in Section 5.4.2.

Once we have formulas for reachability certificates, we can use these to define formulas $\varphi^{(v,c),(v',c')}_{\textit{reach-notest}}(\boldsymbol{x})$, expressing that $(v', c')$ is reachable from $(v, c)$ without taking any zero tests, $\varphi^{(v,c),(v',c')}_{\textit{reach-test}}(\boldsymbol{x})$, expressing reachability via at least one zero test, and $\varphi^{(v,c),(v',c')}_{\textit{reach}}(\boldsymbol{x})$, which is general reachability in 1-CA. The encodings of these formulas are treated in Section 5.4.3.

### 5.4.2 Encoding Reachability Certificates

In this section we give a detailed description of a reduction from the existence of reachability certificates in parameterised 1-CA to the decision problem for ∃PAD. The reduction is nondeterministic in that it relies on a case analysis on the support of each flow. We will need the following definition for type 1 and 2 certificates.

A *decomposition* of a path flow $f$ is an enumeration $v_1, \ldots, v_n$ of the set of vertices with some incoming flow, $\{v \mid f(u, v) > 0 \text{ for some } u \in V\}$, together with a sequence of flows $f_0, f_1, \ldots, f_{n-1}$ such that:

(i) $f_0$ is a path flow from the source of $f$ to $v_1$

(ii) $f_i$ is a path flow from $v_i$ to $v_{i+1}$, for $1 \leq i \leq n-1$

(iii) $f = \sum_{i=0}^{n-1} f_i$

(iv) if $i \leq j$ then no edge pointing into $v_i$ is in the support of $f_j$.

This means that the path flow $f$ corresponds to a path $\gamma$ that is the sequential composition of paths $\gamma_0, \ldots, \gamma_{n-1}$, corresponding to $f_0, \ldots, f_{n-1}$, respectively, and $\gamma_0$ ends in the last occurrence of $v_1$ in $\gamma$, $\gamma_1$ ends in the last occurrence of $v_2$ in $\gamma$, etc.

If $f$ corresponds to more than one path, different paths corresponding to $f$ may induce different decompositions.

**Example 32.** *Consider the graph shown in Figure 5.9. Weights of edges are ignored in this example. Let $f$ be the flow that assigns 1 to the edge $(v_1, v_4)$ and 2 to all other edges. Let $\omega_1 = (v_1, v_2)(v_2, v_1)$ and $\omega_2 = (v_1, v_3)(v_3, v_1)$. The paths*

$$\gamma_1 = \omega_1\omega_1\omega_2\omega_2(v_1, v_4), \qquad \gamma_2 = \omega_1\omega_2\omega_1\omega_2(v_1, v_4),$$
$$\gamma_3 = \omega_2\omega_1\omega_1\omega_2(v_1, v_4), \qquad \gamma_4 = \omega_2\omega_2\omega_1\omega_1(v_1, v_4)$$

*all have Parikh image $f$. $\gamma_2$ and $\gamma_3$ induce the same decomposition with enumeration $v_2, v_3, v_1, v_4$. $\gamma_1$ induces a different decomposition with the same enumeration, and the decomposition corresponding to $\gamma_4$ has the enumeration $v_3, v_2, v_1, v_4$.*

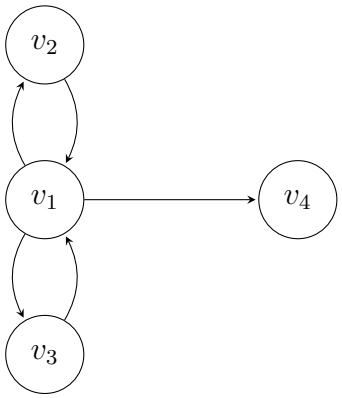


Figure 5.9

**Lemma 33.** *Let $\mathcal{C} = (V, E, X, \lambda)$ be a parameterised 1-CA, let $(v, c)$ be the start configuration and $(v', c')$ the target configuration, and let $f$ be a flow in $G_{\mathcal{C}'}$, where $\mathcal{C}'$ is the result of deleting all zero tests from $\mathcal{C}$. The condition that $f$ is a type 1, 2 or 3 reachability certificate for $(v, c)$ and $(v', c')$ can be encoded in nondeterministic polynomial time (in the size of $\mathcal{C}$ and the bit length of $c$ and $c'$) as a $\exists$PAD formula where the only variables that occur on the left side of a divisibility are those corresponding to the parameters of $\mathcal{C}$.*

*Proof.* Let the action transitions of $\mathcal{C}$ (i.e., those which are not zero tests) be $E = \{e_1, \ldots, e_m\}$, and let the parameters be $X = \{x_1, \ldots, x_r\}$. Let $\boldsymbol{x}$ be the vector $(x_1, \ldots, x_r)$.

Recall that a type-1 certificate of reachability from $(v, c)$ to $(v', c')$ is a path flow $f$ from $v$ to $v'$ that has no positive cycles,

$$\text{weight}(f) = c' - c \tag{5.1}$$

and there is a path $\gamma$ corresponding to $f$ such that

$$\text{drop}(\gamma) \geq -c. \tag{5.2}$$

The condition that $f$ has no positive cycles can be encoded as a polynomial-size formula $\varphi_{\textit{no-pos}}(\boldsymbol{d}, \boldsymbol{x})$ of Presburger arithmetic. Such a formula can be obtained from a static single assignment form of the Bellman-Ford algorithm (after first unrolling all for-loops), introducing new integer variables $d_j^{(i)}$ to represent the distance for node $v_j$ computed by the algorithm after $i$ iterations, and translating assignments to variables into equality constraints.

We nondeterministically choose the support of the flows $f_0, f_1, \ldots, f_{n-1}$ which make up the decomposition of $f$ corresponding to the path $\gamma$ in (5.2). The vertex ordering $v_1, \ldots, v_n$ which belongs to the same decomposition, i.e., the source and target of each flow $f_i$, can be derived from the given supports: if $f_i$ contains a transition to the state $u$ and $f_j$ does not, for all $j > i$, then $v_{i+1} = u$. Since $f$ contains no positive cycles, $\gamma$ cannot contain any positive cycles. So if a vertex $u$ is visited multiple times in $\gamma$, we know that among all prefixes of $\gamma$ ending in $u$, the longest such prefix is the one of minimum weight. This property of $\gamma$ allows us to rewrite (5.2) in terms of the decomposition $f_0, \ldots, f_{n-1}$ of $f$:

$$\bigwedge_{j=0}^{n-1} \left( \sum_{i=0}^{j} \text{weight}(f_i) \geq -c \right). \tag{5.3}$$

Each flow $f_i$ can be represented symbolically using a set of integer variables $\boldsymbol{y}^{(i)} = \{y_1^{(i)}, y_2^{(i)}, \ldots, y_m^{(i)}\}$, where $y_j^{(i)}$ is the flow along edge $e_j$ in $f_i$. Since we have previously guessed the supports of the $f_i$ (in polynomial time), we know which of the $y_j^{(i)}$ are zero. A system of linear constraints $\varphi_{\textit{flow}}^{(i)}$ ensures that $f_i$ is a path flow from the right source to the right target and with the right support. Each constraint has one of the following forms:

(i) $y_j^{(i)} = 0$ if $e_j \notin \text{support}(f_i)$, specifying the support of each $f_i$

(ii) $y_j^{(i)} > 0$ if $e_j \in \text{support}(f_i)$, specifying that a flow cannot be negative

(iii) $\sum_{u \in V} y_{w,u}^{(i)} - \sum_{u \in V} y_{u,w}^{(i)} = 0$ if $w \neq v_i$ and $w \neq v_{i+1}$, specifying flow conservation properties

(iv) $\sum_{u \in V} y^{(i)}_{v_i,u} - \sum_{u \in V} y^{(i)}_{u,v_i} - 1 = 0$, specifying that each $f_i$ is a path flow

where $y^{(i)}_{u,u'} \equiv y^{(i)}_j$ if $e_j$ is a transition from $u$ to $u'$, $y^{(i)}_{u,u'} \equiv 0$ if there is no such transition in $E$, and $v_0 = v$.

The requirements (5.1) and (5.3) can be translated to the formula

$$\varphi_{weights}(\boldsymbol{x}, \boldsymbol{y}) \equiv \bigwedge_{j=0}^{n-1} \left( \sum_{i=0}^{j} W^{(i)}(\boldsymbol{x}, \boldsymbol{y}) \geq -c \right) \wedge \sum_{i=0}^{n-1} W^{(i)}(\boldsymbol{x}, \boldsymbol{y}) = c' - c \tag{5.4}$$

where the term $W^{(i)}(\boldsymbol{x}, \boldsymbol{y}) \equiv \sum_{k=1}^{m} y^{(i)}_k \alpha_k$ with $\lambda(e_k) = \text{add}(\alpha_k)$ for each $k \in \{1, \ldots, m\}$ encodes weight$(f_i)$. The statement that $f$ is a type-1 reachability certificate is then encoded as

$$\exists \boldsymbol{d}\, \varphi_{no\text{-}pos}(\boldsymbol{d}, \boldsymbol{x}) \wedge \exists \boldsymbol{y} \left( \bigwedge_{i=0}^{n-1} \varphi^{(i)}_{flow}(\boldsymbol{y}^{(i)}) \wedge \varphi_{weights}(\boldsymbol{x}, \boldsymbol{y}) \right). \tag{5.5}$$

Note that $\varphi_{weights}$ is not a PAD formula, as it contains products of variables. To construct a PAD formula with the same meaning as (5.5), we first need to eliminate the constraints $\varphi^{(i)}_{flow}$ so that the existentially quantified variables in $\varphi_{weights}$ do not occur in any other subformula. Using Theorem 1 from Section 2.3 we can nondeterministically choose a change of variables from $y^{(i)}_1, \ldots, y^{(i)}_m$ to new positive integer variables $z^{(i)}_1, \ldots, z^{(i)}_{b_i}$. Applying this change of variables to the weight of $f_i$ and rearranging terms gives the formula

$$W^{(i)}_*(\boldsymbol{x}, \boldsymbol{z}) \equiv \sum_{k=1}^{b_i} z^{(i)}_k P^{(i)}_k(\boldsymbol{x}) + Q^{(i)}(\boldsymbol{x}) \tag{5.6}$$

where the $P^{(i)}_k$ and $Q^{(i)}$ are linear polynomials in $\boldsymbol{x}$. This can be substituted into $\varphi_{weights}$ to obtain the formula $\varphi'_{weights}(\boldsymbol{x}, \boldsymbol{z})$.

The statement that $f$ is a type-1 reachability certificate is now expressed as $\exists \boldsymbol{d}\, \varphi_{no\text{-}pos}(\boldsymbol{d}, \boldsymbol{x}) \wedge \exists \boldsymbol{z} \varphi'_{weights}(\boldsymbol{x}, \boldsymbol{z})$ . We can translate $\varphi'_{weights}$ to an equivalent PAD formula $\varphi''_{weights}$ as follows. For every pair of variables $x_j$ and $z^{(i)}_k$, for $1 \leq j \leq r$, $0 \leq i \leq n-1$ and $1 \leq k \leq b_i$, we introduce a new variable $w^{(i)}_{j,k}$, which represents the product of $x_j$ and $z^{(i)}_k$. The terms $z^{(i)}_k P^{(i)}_k(\boldsymbol{x})$ in $\varphi'_{weights}$ can then be rewritten as linear terms in the new variables $\boldsymbol{w}$. Rewriting all of these terms results in the formula $\varphi''_{weights}(\boldsymbol{x}, \boldsymbol{w})$. Now there exist values $\boldsymbol{z}$ that satisfy $\varphi'_{weights}(\boldsymbol{x}, \boldsymbol{z})$ if and only if there exist values $\boldsymbol{w}$ that satisfy $\varphi''_{weights}(\boldsymbol{x}, \boldsymbol{w})$, where every $w^{(i)}_{j,k}$ is required to be divisible by $x_j$ and have the same sign as $x_j$. This results in the $\exists$PAD formula

$$\begin{aligned} \varphi^{(v,c)(v',c')}_{type1}(\boldsymbol{x}) \equiv \exists \boldsymbol{d} \exists \boldsymbol{w} \bigwedge_{j=1}^{r} \bigwedge_{i=0}^{n-1} \bigwedge_{k=1}^{b_i} x_j | w^{(i)}_{j,k} \wedge (x_j > 0 \Leftrightarrow w^{(i)}_{j,k} > 0) \\ \wedge \varphi''_{weights}(\boldsymbol{x}, \boldsymbol{w}) \wedge \varphi_{no\text{-}pos}(\boldsymbol{d}, \boldsymbol{x}). \end{aligned} \tag{5.7}$$

Note that only variables in $\boldsymbol{x}$ occur on the left side of any divisibilities.

A type-2 reachability certificate from $(v, c)$ to $(v', c')$ is equivalent to a type-1 certificate from $(v', c')$ to $(v, c)$ in the skew transpose of $G_{\mathcal{C}}$, that is, the weighted graph obtained from $G_{\mathcal{C}}$ by multiplying every weight by $-1$ and reversing the direction of every edge. The resulting PAD formula $\varphi_{type2}$ is of the same form as $\varphi_{type1}$.

A type-3 reachability certificate from $(v, c)$ to $(v', c')$ is a path flow $f$ from $v$ to $v'$ with weight$(f) = c' - c$ such that there is a positive cycle in $f$ that can be taken from $(v, c)$ and there is a negative cycle in $f$ that can be taken to $(v', c')$. To get a translation to ∃PAD in nondeterministic polynomial time, we can simply guess a positive cycle $\omega_1 = e_1^{(1)} \dots e_k^{(1)}$ that can be taken from $(v, c)$ and a negative cycle $\omega_2 = e_l^{(2)} \dots e_1^{(2)}$ that can be taken to $(v', c')$. Let

$$\begin{aligned}\varphi_{cycles}(\boldsymbol{x}) \equiv \operatorname{start}(e_1^{(1)}) = \operatorname{end}(e_k^{(1)}) = v \wedge \operatorname{start}(e_l^{(2)}) = \operatorname{end}(e_1^{(2)}) = v' \wedge \\ \bigwedge_{j=1}^{k} \left( \sum_{i=1}^{j} \beta_i \geq -c \right) \wedge \sum_{i=1}^{k} \beta_i > 0 \wedge \\ \bigwedge_{j=1}^{l} \left( \sum_{i=1}^{j} -\beta'_i \geq -c' \right) \wedge \sum_{i=1}^{l} \beta'_i < 0 \end{aligned} \tag{5.8}$$

where $\lambda(e_i^{(1)}) = \operatorname{add}(\beta_i)$ and $\lambda(e^{(2)}) = \operatorname{add}(\beta'_i)$ for $1 \leq i \leq m$. A type-3 certificate from $(v, c)$ to $(v', c')$ can then be specified as follows:

$$\varphi_{type3}^{(v,c),(v',c')}(\boldsymbol{x}) \equiv \exists \boldsymbol{y}\, \varphi_{cycles}(\boldsymbol{x}) \wedge \sum_{k=1}^{m} y_i \alpha_k = c' - c \wedge \varphi_{flow}(\boldsymbol{y}), \tag{5.9}$$

where $\varphi_{flow}(\boldsymbol{y})$ is a linear system of equations similar to the $\varphi_{flow}^{(i)}$ above, ensuring that $f$ is a path flow from $v$ to $v'$ with the right support, and $\lambda(e_k) = \operatorname{add}(\alpha_k)$ for $1 \leq i \leq m$. $y_i$ simply represents $f(e_i)$, as there is no flow decomposition to consider in this case. We do not need constraints to specify that drop$(f) \geq -c$, as this is implied by the existence of the positive and negative cycle in the graph, see [HKOW09, Proposition 5]. Again, using a change of variables, the formula $\varphi_{flow}$ can be eliminated, and the resulting formula can be translated to an equivalent ∃PAD formula. Again, only variables in $\boldsymbol{x}$ occur on the left side of a divisibility. □

### 5.4.3 Encoding Full Reachability

**Proposition 34.** *Reachability without zero tests, reachability via zero tests, and general reachability in 1-CA can all be encoded in nondeterministic polynomial time as formulas of ∃PAD with free variables $\boldsymbol{x}$ corresponding to the parameters of the 1-CA, where $\boldsymbol{x}$ are the only variables that occur on the left side of any divisibilities.*

*Proof.* We first define $\varphi^{(v,c),(v',c')}_{\textit{reach-notest}}(\boldsymbol{x})$, which expresses that there exists a computation from $(v, c)$ to $(v', c')$ that does not include any zero test transitions. Combining Proposition 31 with the results from the previous section, we get

$$\varphi^{(v,c),(v',c')}_{\textit{reach-notest}}(\boldsymbol{x}) \equiv \exists k \exists k' \bigvee_{u \in V} \bigvee_{u' \in V} \left( \varphi^{(v,c),(u,k)}_{\textit{type1}}(\boldsymbol{x}) \wedge \varphi^{(u,k),(u',k')}_{\textit{type2}}(\boldsymbol{x}) \wedge \varphi^{(u',k'),(v',c')}_{\textit{type3}}(\boldsymbol{x}) \right).$$

$\varphi_{\textit{reach-test}}$, which expresses that a configuration is reachable from another configuration via at least one zero test, can then be defined by nondeterministically choosing an ordering of the zero tests that are taken, and encoding the corresponding computation as a conjunction of $\varphi_{\textit{reach-notest}}$ formulas.

Finally, general reachability can be expressed as

$$\varphi^{(v,c),(v',c')}_{\textit{reach}}(\boldsymbol{x}) = \varphi^{(v,c),(v',c')}_{\textit{reach-test}}(\boldsymbol{x}) \vee \varphi^{(v,c),(v',c')}_{\textit{reach-notest}}(\boldsymbol{x}).$$

For a given parameterised 1-CA $\mathcal{C} = (V, E, X, \lambda)$ with start configuration $(v, c)$ and target configuration $(v', c')$, there exist values for the parameters $X$ such that there is a computation from $(v, c)$ to $(v', c')$ if and only if $\exists \boldsymbol{x} \varphi^{(v,c),(v',c')}_{\textit{reach}}(\boldsymbol{x})$ evaluates to true. □

**Corollary 35.** *The reachability problem for parameterised 1-CA is in* ***NEXPTIME****.*

This follows from Corollary 26 from Section 4.4, which states our new result that the decision problem for ∃PAD is in **NEXPTIME**. By Proposition 29, we can conclude that reachability in parametric 1-CA and the decision problem for ∃PAD are equivalent in terms of complexity, meaning that any improved complexity bounds for one problem would give us the same bounds for the other.

## 5.5 Encoding Synthesis Problems

### 5.5.1 Encoding Universal Co-Büchi Synthesis

The negation of the universal co-Büchi synthesis problem asks, given a parameterised 1-CA $\mathcal{C} = (V, E, X, \lambda)$ with a set of accepting states $V_{Acc} \subseteq V$ and an initial configuration $(v, c)$, whether for all possible parameter values, there is some infinite computation starting in $(v, c)$ that visits some accepting state infinitely often. This can be expressed in terms of the reachability formulas from the previous section.

**Proposition 36.** *Given a parameterised 1-CA $\mathcal{C} = (V, E, X, \lambda)$ with a set of accepting states $V_{Acc} \subseteq V$ and an initial configuration $(v, c)$, the negation of the universal*

*co-Büchi synthesis problem for $\mathcal{C}$ and $(v, c)$ can be encoded in (deterministic) singly exponential time as a formula of PAD with one quantifier alternation, where existentially quantified variables are only allowed to occur on the right-hand side of any divisibility predicate.*

*Proof.* We need to use the formulas $\varphi_{reach}$, $\varphi_{reach\text{-}test}$ and $\varphi_{reach\text{-}notest}$ to express that it is possible to reach some configuration $(u, k)$ from $(v, c)$, where $u$ is accepting, and from where a cycle can be taken back to $u$ infinitely many times. We need to distinguish between two cases: either the cycle contains a zero test, or it does not. In the first case it needs to return to the exact configuration $(u, k)$, to guarantee that the zero test(s) could be taken again infinitely many times. In the second case it suffices to assert that the cycle is not negative so it can be taken infinitely many times without the counter ever becoming negative. Quantifying over all parameter values, we get the following formula for the negation of Büchi acceptance synthesis:

$$\forall \boldsymbol{x} \exists k \exists k' \bigvee_{u \in V_{Acc}} (k \leq k') \wedge \varphi_{reach}^{(v,c),(u,k)}(\boldsymbol{x}) \wedge \left( \varphi_{reach\text{-}test}^{(u,k),(u,k)}(\boldsymbol{x}) \vee \varphi_{reach\text{-}notest}^{(u,k),(u,k')}(\boldsymbol{x}) \right). \quad (5.10)$$

Note that the reachability formulas are purely existential, and that the universally quantified variables $\boldsymbol{x}$ are the only ones that occur on the left side of any divisibilities. Moreover, the reachability formulas can be built in deterministic singly exponential time following the same steps as in the previous section, but using disjunctions instead of nondeterminism. □

**Corollary 37.** *Decidability of the fragment of PAD with one quantifier alternation and with existential quantifiers restricted to occur only on the right side of any divisibilities would imply decidability for the universal co-Büchi synthesis problem.*

Recall that the decidability result for this fragment of PAD in [BI05] is currently under review.

### 5.5.2 Reducing LTL Synthesis to Universal Co-Büchi Synthesis

Recall Theorem 4 from Section 2.5.3, which states that every LTL formula $\varphi$ over a set of propositions $P$ can be translated in polynomial space to a generalised Büchi automaton $A$ of at most exponential size such that $\varphi$ and $A$ accept the same language. Recall also Lemma 3 from Section 2.5.2, which states that every generalised Büchi automaton is equivalent to a Büchi automaton of polynomial size.

**Theorem 38.** *There is a reduction from the negation of LTL synthesis to the negation of universal co-Büchi synthesis.*

*Proof.* We use an approach similar to standard LTL model checking algorithms for Kripke structures to reduce the negation of the LTL synthesis problem to the negation of the universal co-Büchi synthesis problem. The negation of LTL synthesis asks, for a given parameterised 1-CA $\mathcal{C}$ and LTL formula $\varphi$, if for all possible values for the parameters of $\mathcal{C}$, there exists a computation in $\mathcal{C}$ along which $\neg\varphi$ holds.

Given a parameterised 1-CA $\mathcal{C} = (V, E, X, \lambda)$ with an initial configuration $(v, c)$ and with $\mu : E \to 2^P$ labelling each transition in $E$ by a subset of propositions from $P$, and given an LTL formula $\varphi$ over $P$, the aim is to construct a parameterised 1-CA $\mathcal{C}_{\neg\varphi} = (V', E', X, \lambda')$ with a set of accepting states $V_{acc} \subseteq V'$ such that for all values of $X$, there is a computation in $\mathcal{C}$ that satisfies $\neg\varphi$ if and only if there is a computation in $\mathcal{C}_{\neg\varphi}$ that visits some final state infinitely often.

We first build a generalised Büchi automaton $A^{(g)}_{\neg\varphi} = (V^{(g)}_A, 2^P, E^{(g)}_A, V^{(g)}_0, \mathcal{F}^{(g)}_A)$ such that $w \in L(A)$ if and only if $w, 0 \vDash \neg\varphi$, using Theorem 4. Then, using Lemma 3, we construct an equivalent Büchi automaton $A_{\neg\varphi} = (V_A, 2^P, E_A, V_0, F_A)$. We then build $\mathcal{C}_{\neg\varphi}$ as the product automaton of $\mathcal{C}$ and $A_{\neg\varphi}$ as follows:

- $V' = V \times V_A$
- $((v_1, v_{a1}), (v_2, v_{a2})) \in E'$ iff $(v_1, v_2) \in E$ and $(v_{a1}, \mu(v_1, v_2), v_{a2}) \in E_A$
- $\lambda'((v_1, v_{a1}), (v_2, v_{a2})) = \lambda(v_1, v_2)$
- $(v_1, v_{a1}) \in V_{acc}$ iff $v_{a1} \in F_A$

The possible initial configurations of $\mathcal{C}_{\neg\varphi}$ are $\{((v, v_a), c) : v_a \in V_0\}$. The case of more than one initial configuration in the universal co-Büchi synthesis problem is easily handled by using disjunctions of reachability formulas in (5.10). Now the negation of the condition of the LTL synthesis problem for $\mathcal{C}$ and $\varphi$ evaluates to true if and only if the negation of the condition of the Büchi synthesis problem for $\mathcal{C}_{\neg\varphi}$ evaluates to true. □

## 5.6 A Lower Bound for Universal Co-Büchi Synthesis

In this section we will establish a $\mathbf{NP^{NP}}$ lower complexity bound for the universal co-Büchi synthesis problem by a reduction from a generalisation of subset sum with one quantifier alternation, which takes as input two sets of integers $A = \{a_1, a_2, \ldots, a_m\}$

and $B = \{b_1, b_2, \ldots, b_n\}$, as well as an integer goal $g$. The problem is to decide whether the following statement holds:

$$\exists S \subseteq A \, \forall T \subseteq B \left( \sum_{s \in S} s + \sum_{t \in T} t \neq g \right). \tag{5.11}$$

That is, is there a selection of elements from $A$ such that for all possible selections of elements from $B$, the sum of all selected elements does not equal the goal. This problem is known to be $\mathbf{NP}^{\mathbf{NP}}$-complete [BKL+97].

**Proposition 39.** *There is a polynomial-time reduction from the generalised subset sum problem with one quantifier alternation to the universal co-Büchi synthesis problem for parameterised 1-CA.*

*Proof.* Given $A$, $B$ and $g$ as above, we construct a 1-CA $\mathcal{C} = (V, E, X, \lambda)$ with a set of parameters $X = \{x_1, x_2, \ldots, x_m\}$, an accepting subset of states $V_{acc} \subseteq V$ and an initial configuration $(v_0, 0)$ such that (5.11) holds if and only if there is a way of substituting integer values for the parameters $X$ that forces all computations from $(v_0, 0)$ to visit states in $V_{acc}$ only finitely many times. The parameters $X$ encode the choice of elements in the set $S$, while the elements in $T$ will be represented by computations that can be taken in $\mathcal{C}$.

If for any $1 \leq i \leq m$, $x_i$ is not either 0 or $a_i$, there is a computation from the initial configuration $(v_0, 0)$ to a "bad state", i.e., a state from where a cycle can be taken infinitely many times that contains accepting states. So for all computations to be guaranteed to never enter such a cycle, the parameters need to be assigned values such that each $x_i$ is either 0 or $a_i$. Intuitively $x_i = a_i$ means that the element $a_i$ is included in the set $S$, and $x_i = 0$ means that $a_i$ is not included in $S$. $\mathcal{C}$ will then ensure that for any such values for the parameters, a bad state can be reached if and only if there is a subset of $B$ whose elements, combined with the $x_i$, sum to $g$.

Without loss of generality, we can assume that the elements of the sets $A$ and $B$ are sorted in decreasing order, and that $g \geq 0$. Let $\{a_1, a_2, \ldots, a_k\}$ be the positive elements of $A$, and let $\{b_1, b_2, \ldots, b_l\}$ be the positive elements of $B$.

Figure 5.10 shows a sub-automaton that allows a computation from $(v_{i-1}, 0)$ to one of the states $u_i$ and $u_i'$ if and only if $x_i \neq 0$ and $x_i \neq a_i$, assuming $a_i$ is positive. If $x_i \leq 0$, it can take the first downwards transition, and only if $x_i < 0$ it can take the second transition to $u_i'$. If $x_i \geq 0$ then the transition to the right can be taken, storing the value of $x_i$ as the new counter value. The following two downwards transitions can only be taken if $x_i > 0$. From here, some positive integer is either added or subtracted

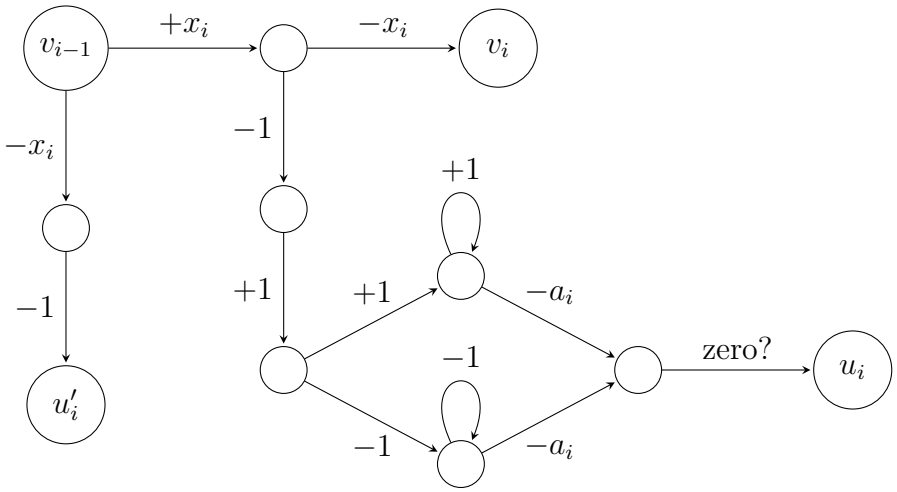


Figure 5.10: The sub-automaton $\mathcal{C}_i$ for positive $a_i$. One of $u_i$ and $u'_i$ is reachable from $(v_{i-1}, 0)$ iff $x_i \neq 0$ and $x_i \neq a_i$.

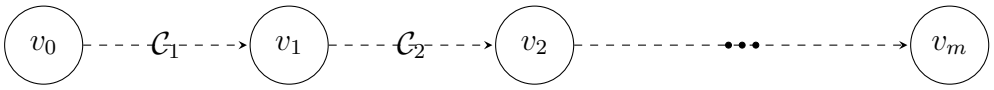


Figure 5.11: The series of automata $\mathcal{C}_1, \ldots, \mathcal{C}_m$.

to reach the value $a_i$. Clearly, if $x_i = a_i$ then the zero test to $u_i$ can never be taken. We will call this sub-automaton $\mathcal{C}_i$. If $a_i$ is negative, then we can simply reverse the sign of the labels of the action transitions involving $x_i$ and $a_i$ to obtain an automaton with the same effect.

We then put these $m$ automata in series, as shown in Figure 5.11. On the path $v_0 \ldots v_m$, as the initial configuration was $(v_0, 0)$, the counter value will be 0 in each state $v_i$ for $0 \leq i \leq m$.

From $(v_m, 0)$, there is a series of $k$ transitions that load the sum of the $x_i$ that correspond to positive elements of $A$ onto the counter. As can be seen in Figure 5.12, this is followed by $l$ triangle shaped sub-automata that allow the options of either

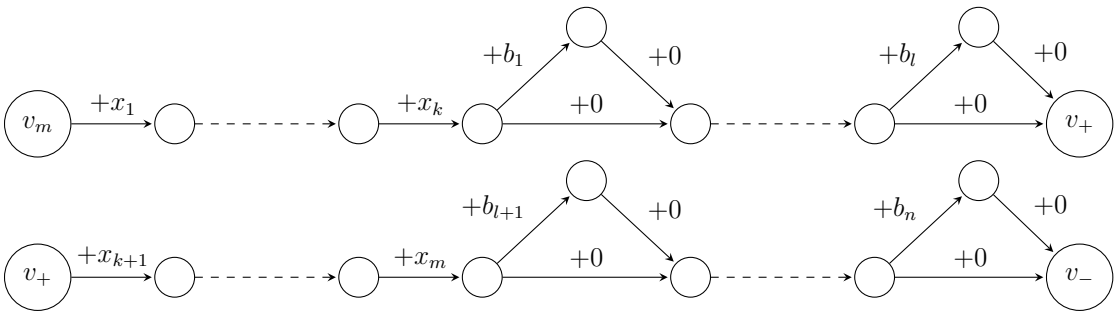


Figure 5.12: Adding positive and negative values to the counter.

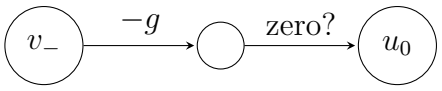


Figure 5.13: The final "decision part" of the automaton.

adding $b_i$ or 0, for $1 \leq i \leq l$. This pattern is repeated for the negative values in $A$ and $B$. The state $v_-$ has a series of two transitions to $u_0$, see Figure 5.13. Finally, we complete the 1-CA by adding a loop labelled add(0) on every state. For the two states in each $\mathcal{C}_i$ that already have a loop, we add an additional add(0) transition from them to a new state with a add(0) loop.

We now set $V_{acc} = \{u_0, u_1, \ldots, u_m, u'_1, \ldots, u'_m\}$. If any of the $x_i$ are assigned a value different from 0 and $a_i$, one of the bad states $u_i$ and $u'_i$ is reachable from $(v_0, 0)$. If every $x_i$ is either 0 or $a_i$, then the only bad state that could still be reached is $u_0$. This will happen if for some subset $T \subseteq B$, the sum of all the $x_i$ and all the elements of $T$ equals $g$. □

# Chapter 6

# One-Counter Automata with Disequality Tests

## 6.1 Introduction

Runs of infinite-state machines, such as counter automata, can naturally be seen as *data words*, that is, sequences in which each position is labelled by a letter from a finite alphabet and another letter (often called the datum) from an infinite alphabet. Freeze LTL is an extension of Linear Temporal Logic with registers (or variables) and a binding mechanism, which has been introduced to specify properties of data words [DL06, DLN05, Fre03, LP05]. The registers allow to compare data at different positions along the same computation.

An example of a freeze LTL formula is

$$\mathsf{F}(v \wedge \downarrow_r \mathsf{XF}(v \wedge \uparrow_r)) \,. \tag{6.1}$$

Evaluated on a run of a one-counter automaton, this formula is true if and only if there are at least two different positions in the run which both have control state $v$ and the same counter value. Intuitively the operator $\downarrow_r$ binds the current counter value to register $r$, while the operator $\uparrow_r$ tests whether the current counter value is equal to the content of register $r$.

It is known that the model checking problem for Freeze LTL on classical one-counter automata (i.e., non-parameterised one-counter automata as defined in Chapter 5), is undecidable, but **PSPACE**-complete if one restricts to *deterministic* one-counter automata [DLS08]. Rather than restricting the class of one-counter automata, one can seek to identify decidable syntactic fragments of Freeze LTL. This approach was pursued in [DS10], which studied the *flat* fragment of Freeze LTL. The flatness condition places restrictions on the occurrence of the binding construct $\downarrow_r$ in relation

to the until operator (see Section 6.2.2 for details). For example, in a flat formula in negation normal form the binding operator $\downarrow_r$ can occur within the scope of $\mathsf{F}$ but not $\mathsf{G}$. (Thus formula (6.1) is flat.) The flatness restriction for Freeze LTL has a similar flavour to the flatness restriction for constraint LTL [CC00] and for Metric Temporal Logic [BMOW08].

Demri and Sangnier reduced the model checking problem for flat Freeze LTL on classical one-counter automata to a repeated reachability problem for one-counter automata with parameterised equality and disequality tests, and left the decidability of the latter as an open problem. The main technical contribution of this chapter is to show decidability of this problem by reduction to the decision problem for Presburger arithmetic, thus also obtaining decidability of the model checking problem for flat Freeze LTL formulas on (non-parameterised) one-counter automata.

## 6.2 Preliminaries

### 6.2.1 One-Counter Automata with Equality and Disequality Tests

Recall that in this chapter we will work with a more general version of one-counter automata compared to the previous chapter, and we will redefine the terms *one-counter automaton*, *configuration* and *computation* for this purpose.

A *one-counter automaton* (1-CA) is a tuple $\mathcal{C} = (V, E, \lambda, \tau)$, where $V$ is a finite set of *states*, $E \subseteq V \times V$ is a finite set of *edges* between states, $\lambda : E \to Op$ labels each edge with an element from $Op = \{\mathrm{add}(a) : a \in \mathbb{Z}\} \cup \{\mathrm{eq}(a) : a \in \mathbb{N}\}$, and $\tau : V \to 2^{\mathbb{N}}$ maps each state $v$ to a finite set $\tau(v)$ of *invalid counter values* at state $v$. Intuitively the operation $\mathrm{add}(a)$ adds $a$ to the counter and $\mathrm{eq}(a)$ tests the counter for equality with $a$. The association of invalid counter values with each state can be seen as a type of disequality test. This feature will be required for our treatment of freeze LTL.

For any edge $e = (v, v')$ with $\lambda(e) = \mathrm{op}(a)$, define $\mathrm{start}(e) = v$, $\mathrm{end}(e) = v'$, and $\mathrm{weight}(e) = a$ if $\mathrm{op} = \mathrm{add}$, $\mathrm{weight}(e) = 0$ if $\mathrm{op} = \mathrm{eq}$. A *path* $\gamma$ is a finite word on the alphabet $E$: $\gamma = e_1 \cdots e_n$ such that $\mathrm{end}(e_i) = \mathrm{start}(e_{i+1})$ for every $i$. Definitions of simple paths, weights of paths, cycles, simple cycles, etc. are analogous with those for weighted graphs, see Section 2.4.1.

A *configuration* of a 1-CA $\mathcal{C} = (V, E, \lambda, \tau)$ is a pair $(v, c)$ with $v \in V$ and $c \in \mathbb{Z}$. Intuitively, $(v, c)$ corresponds to the situation where the 1-CA is in state $v$ with counter value $c$. Configurations $(v, c)$ with $c \geq 0$ are called *valid*, otherwise they are *invalid*. The relation $E$ between states with guards $\lambda$ and $\tau$ induces an unlabelled transition

relation between configurations: for any two configurations $(v, c)$ and $(v', c')$, there is a transition $(v, c) \longrightarrow (v', c')$ if and only if there is an edge $e \in E$ with $\lambda(e) = \mathrm{op}(a)$ for some $a$ such that $\mathrm{start}(e) = v$, $\mathrm{end}(e) = v'$, and $\mathrm{weight}(e) = c' - c$. We will sometimes write $(v, c) \xrightarrow{e} (v', c')$ for such a transition. The transition is *valid* if both $(v, c)$ and $(v', c')$ are valid configurations (i.e., $c, c' \geq 0$) and $c \notin \tau(v)$, and also $c = a$ if $\mathrm{op} = \mathrm{eq}$. Otherwise it is *invalid*.

A *computation* $\pi$ is a (finite or infinite) sequence of transitions:

$$\pi = (v_1, c_1) \longrightarrow (v_2, c_2) \longrightarrow (v_3, c_3) \longrightarrow \cdots$$

We write $|\pi|$ for the length of $\pi$. If $(v_1, c_1) \xrightarrow{e_1} (v_2, c_2) \xrightarrow{e_2} \cdots \xrightarrow{e_{n-1}} (v_n, c_n)$ is a finite computation, we will also write it as $(v_1, c_1) \xrightarrow{\gamma}{}^* (v_n, c_n)$, where $\gamma = e_1 e_2 \cdots e_{n-1}$, or simply $(v_1, c_1) \longrightarrow^* (v_n, c_n)$. A computation $\pi$ is *valid* if all transitions in the sequence are valid, otherwise it is *invalid*. If $\pi$ is invalid, an *obstruction* is a configuration $(v_i, c_i)$ such that $(v_i, c_i) \longrightarrow (v_{i+1}, c_{i+1})$ is an invalid transition, or, if $\pi$ is of finite length $n - 1$, $i = n$ and $c_i < 0$.

Given a path $\gamma$ and a counter value $c \in \mathbb{N}$, the *path computation* $\gamma(c)$ is the (finite) computation starting at $(\mathrm{start}(\gamma), c)$ and following the sequence of transitions that correspond to the edges in $\gamma$.

A *one-counter automaton with parameterised tests* is a tuple $(V, E, X, \lambda, \tau)$, where $V$, $E$ and $\lambda$ are defined as before for 1-CA, $X$ is a set of nonnegative integer variables, $Op = \{\mathrm{add}(a) : a \in \mathbb{Z}\} \cup \{\mathrm{eq}(a), \mathrm{eq}(x) : a \in \mathbb{N}, x \in X\}$ includes parameterised equality tests (but not parameterised actions), and $\tau : V \to 2^{\mathbb{N} \cup X}$ includes parameterised disequality tests. Note that $\tau(v)$ is still required to be finite for each $v \in V$.

For a given 1-CA $\mathcal{C} = (V, E, \lambda, \tau)$, an initial configuration $(v, c)$ and a target configuration $(v', c')$, the *reachability* problem asks if there is a valid computation from $(v, c)$ to $(v', c')$. When $\mathcal{C}$ has sets $F_1, \ldots, F_n \subseteq V$ of final states and an initial configuration $(v, c)$, the *generalised repeated control-state reachability* problem asks if there is a valid infinite computation from $(v, c)$ which visits at least one state in each $F_i$ infinitely often.

For a given 1-CA $\mathcal{C} = (V, E, X, \lambda, \tau)$ with parameterised tests with initial configuration $(v, c)$ and target configuration $(v', c')$, the *reachability* problem asks if there exist values for the parameters such that there is a computation from $(v, c)$ to $(v', c')$. Similarly, in the case where $\mathcal{C}$ has sets $F_1, \ldots, F_n \subseteq V$ of final states and an initial configuration $(v, c)$, the *generalised repeated control-state reachability* problem asks if there exist values for the parameters such that substituting these values satisfies the generalised repeated reachability condition above.

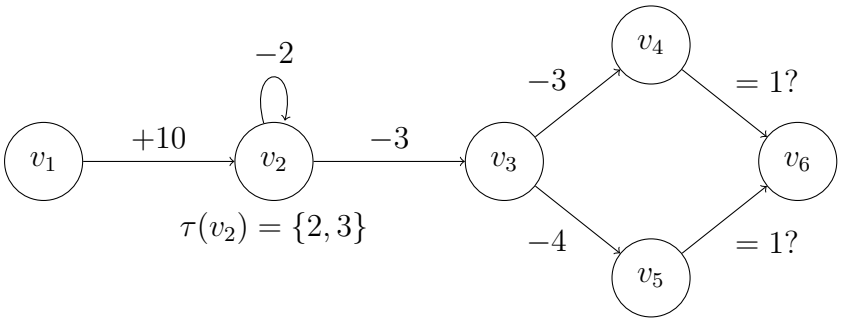


Figure 6.1: A simple 1-CA including a disequality test on the state $v_2$ and equality tests on the transitions $(v_4, v_6)$ and $(v_5, v_6)$.

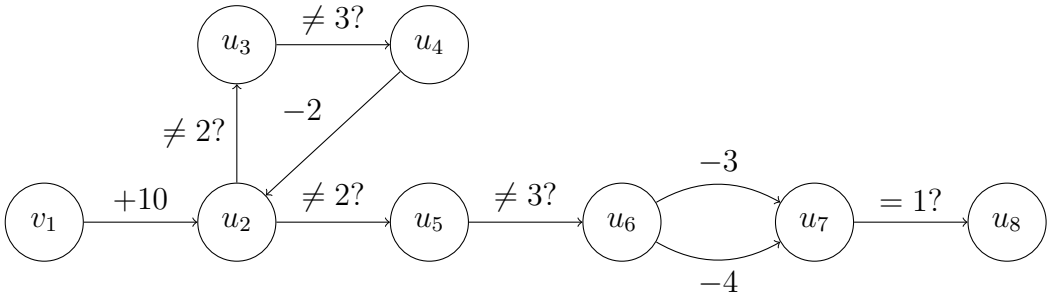


Figure 6.2: An automaton with disequality tests on transitions rather than states and with multiple edges which is equivalent to the one in Figure 6.1.

Note that while we have defined equality tests ($\lambda$) to be on transitions of a 1-CA, such that the counter is required to have a certain value for a transition with an equality test to be valid, disequality tests ($\tau$) are on the states of the 1-CA, such that the counter is required to not be contained in a certain set of values for any transition out of the current state to be valid. It may seem more intuitive to define both types of test on transitions instead, and in fact this is how 1-CA were defined in [DS10], where the above reachability problems for 1-CA with parameterised tests were stated as open. The same paper also allows multiple edges between any two states of a 1-CA. The reason why we chose to work with a model that has disequality tests on states and only at most one edge between any two given states is that it simplifies some of the following proofs.

It is easy to see that the type of 1-CA in this chapter is equivalent to the one defined in [DS10], so that an algorithm for reachability for one of them also solves the same problem for the other. For example, consider the 1-CA in Figure 6.1 (in graphs, we write $+a$ for $\mathrm{add}(a)$ and $= a?$ for $\mathrm{eq}(a)$) and the 1-CA in Figure 6.2 which has disequality tests defined on transitions rather than states as well as multiple edges between $u_6$ and $u_7$. Clearly $(v_1, 0) \longrightarrow^* (v_6, 1)$ if and only if $(u_1, 0) \longrightarrow^* (u_8, 1)$.

### 6.2.2 Model Checking Freeze LTL on One-Counter Automata

Recall the definition of linear temporal logic (LTL) from Section 2.5.1. *Freeze LTL* [DLS08] is an extension of LTL that can be used to specify properties of *data words*. A data word is a (finite or infinite) sequence of symbols, each of which consists of a letter from a finite alphabet and another letter, often referred to as a *datum*, from an infinite alphabet.

Freeze LTL is one of a variety of formalisms that arise by augmenting a temporal or modal logic with variable binding. Given a finite alphabet $\Sigma$ and set of *registers* $R$, the formulas of Freeze LTL are given by the following grammar

$$\varphi \quad ::= \quad a \quad | \quad \uparrow_r \quad | \quad \neg\varphi \quad | \quad \varphi \vee \varphi \quad | \quad \mathsf{X}\,\varphi \quad | \quad \varphi\,\mathsf{U}\,\varphi \quad | \quad \varphi\,\mathsf{R}\,\varphi \quad | \quad \downarrow_r \varphi\,,$$

where $a \in \Sigma$ and $r \in R$. In addition to standard LTL, Freeze LTL contains an atomic freeze formula $\uparrow_r$ and a freeze operator $\downarrow_r$. We write $\mathrm{LTL}^{\downarrow}$ for the set of formulas of Freeze LTL. A *sentence* is a formula in which each occurrence of a subformula $\uparrow_r$ is in the scope of an operator $\downarrow_r$ (for the same register $r$).

In general, formulas of $\mathrm{LTL}^{\downarrow}$ are interpreted over data words which have $\Sigma$ as their finite alphabet and an arbitary infinite alphabet. We are interested in a particular kind of data word, namely those arising from valid computations of 1-CA, and we directly define the semantics of $\mathrm{LTL}^{\downarrow}$ over such computations, assuming that the alphabet $\Sigma$ is the set of control locations of the 1-CA and that the infinite alphabet for data words is $\mathbb{N}$. In this context $\downarrow_r$ can be seen as a binding construct that stores in register $r$ the counter value at the current position in a computation, while $\uparrow_r$ tests whether the counter value at the current position is equal to the content of register $r$. Formally, define a *register valuation* to be a partial function $f : R \to \mathbb{N}$ and consider a valid infinite computation

$$\pi = (v_1, c_1) \longrightarrow (v_2, c_2) \longrightarrow (v_3, c_3) \longrightarrow \cdots$$

of a 1-CA $\mathcal{C}$. We define a satisfaction relation $\pi, i \vDash_f \varphi$ specifying when an $\mathrm{LTL}^{\downarrow}$

formula $\varphi$ is satisfied at position $i$ in $\pi$ under valuation $f$:

$$\begin{aligned}
\pi, i \vDash_f a &\overset{\text{def}}{\Leftrightarrow} v_i = a \\
\pi, i \vDash_f \uparrow_r &\overset{\text{def}}{\Leftrightarrow} c_i = f(r) \\
\pi, i \vDash_f \neg\varphi &\overset{\text{def}}{\Leftrightarrow} \pi, i \nvDash_f \varphi \\
\pi, i \vDash_f \varphi_1 \vee \varphi_2 &\overset{\text{def}}{\Leftrightarrow} \pi, i \vDash_f \varphi_1 \text{ or } \pi, i \vDash_f \varphi_2 \\
\pi, i \vDash_f \mathsf{X}\varphi &\overset{\text{def}}{\Leftrightarrow} \pi, i+1 \vDash_f \varphi \\
\pi, i \vDash_f \varphi_1 \,\mathsf{U}\, \varphi_2 &\overset{\text{def}}{\Leftrightarrow} \pi, j \vDash_f \varphi_2 \text{ for some } j \geq i \text{ and } \pi, k \vDash_f \varphi_1 \text{ for all } i \leq k < j \\
\pi, i \vDash_f \varphi_1 \,\mathsf{R}\, \varphi_2 &\overset{\text{def}}{\Leftrightarrow} \pi, j \vDash_f \varphi_2 \text{ for all } j \geq i, \\
&\quad \text{or for some } j', \pi, j' \vDash_f \varphi_1 \text{ and for all } i \leq k \leq j', \pi, k \vDash \varphi_2 \\
\pi, i \vDash_f \downarrow_r \varphi &\overset{\text{def}}{\Leftrightarrow} \pi, i \vDash_{f[r \mapsto c_i]} \varphi
\end{aligned}$$

where $f[r \mapsto c]$ is the function that maps $r$ to $c$ and is otherwise equal to $f$. Note that the clauses for the Boolean and LTL connectives are defined in the same way as for standard LTL. We also use the same abbreviations ($\mathsf{F}$, $\mathsf{G}$, ...) as for LTL.

An occurrence of a subformula in a LTL$^\downarrow$ formula is *positive* if it lies within the scope of an even number of negations, otherwise it is *negative*. The *flat fragment* of LTL$^\downarrow$ is the set of LTL$^\downarrow$ formulas such that in every positive occurrence of a subformula $\varphi_1 \,\mathsf{U}\, \varphi_2$ (resp. $\varphi_2 \,\mathsf{R}\, \varphi_1$), the binding operator $\downarrow_r$ does not appear in $\varphi_1$, and in every negative occurrence of such a subformula, $\downarrow_r$ does not appear in $\varphi_2$ for any register $r$.

The negation of many natural LTL$^\downarrow$ specifications yield flat formulas. For example, consider the response property $\mathsf{G}(\downarrow_r (\text{req} \rightarrow \mathsf{F}(\text{serve} \wedge \uparrow_r)))$, expressing that every request is followed by a serve with the same associated ticket. The negation of this formula is equivalent to $\mathsf{F}(\downarrow_r (\text{req} \wedge \mathsf{G}(\neg\text{serve} \vee \neg\uparrow_r)))$. The latter is easily seen to be flat after rewriting to the core LTL$^\downarrow$ language with only the $\mathsf{U}$ temporal operator.

The main subject of this chapter is the decidability of the following model checking problem: given a 1-CA $\mathcal{C}$, a valid configuration $(v, c)$ of $\mathcal{C}$, and a flat sentence $\varphi \in \text{LTL}^\downarrow$, does there exist a valid infinite computation $\pi$ of $\mathcal{C}$, starting at $(v, c)$, such that $\pi, 1 \vDash_\emptyset \varphi$? Note that, following [DS10], we have given an existential formulation of the model checking problem. The problem above is equivalent to asking whether $\neg\varphi$ holds along all valid infinite computations starting at $(v, c)$.

The model checking problem for flat LTL$^\downarrow$ on 1-CA was reduced to the generalised repeated reachability problem for 1-CA with parameterised tests in [DS10, Theorem 15]. The idea of the reduction is, given a 1-CA $\mathcal{C}$ and a flat LTL$^\downarrow$ sentence $\varphi$ in negation normal form, to construct a 1-CA with parameterised tests which is the product of $\mathcal{C}$ and $\varphi$. This product automaton includes a parameter $x_r$ for each register $r$ that is mentioned in a subformula of $\varphi$ of type $\downarrow_r \varphi'$. This is where the restriction to the flat

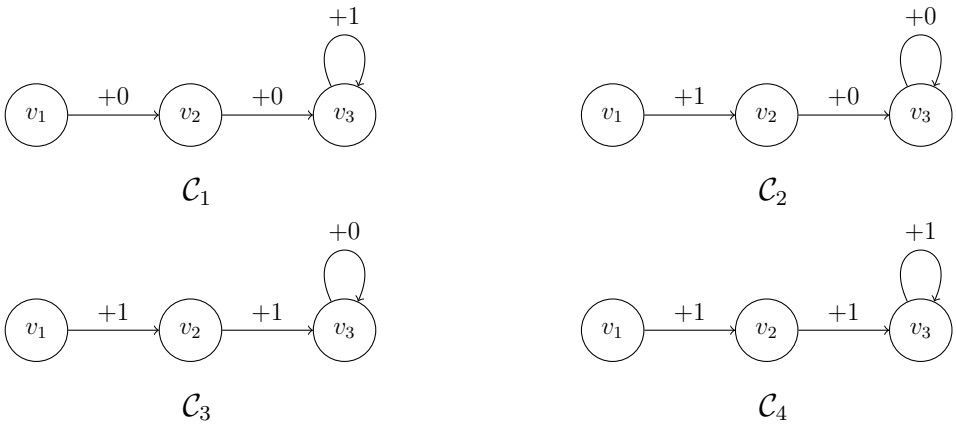


Figure 6.3: The automata considered in Example 40

fragment of LTL$^{\downarrow}$ is crucial, since it allows us to assume, without loss of generality, that the value stored in a register is never overwritten along any computation of $\mathcal{C}$, so that it can be represented by precisely one parameter. An occurrence of the binding operator $\downarrow_r$ in $\varphi$ is represented in the product automaton by an equality test $\mathrm{eq}(x_r)$. A positive occurrence of a formula of the type $\uparrow_r$ is likewise represented by an equality test $\mathrm{eq}(x_r)$, while a negative occurrence of such a subformula is represented by a disequality test $\tau(v_r) = \{x_r\}$.

**Example 40.** *Consider the flat LTL$^{\downarrow}$ formula*

$$\varphi \equiv \mathit{true}\ \mathsf{U}\ (\downarrow_r \mathsf{X}(\uparrow_r \wedge \mathsf{X} \uparrow_r))$$

*and the 1-CA $\mathcal{C}_1$ in Figure 6.3. We will assume that for every 1-CA in this example, the initial counter value is c. To check if $\mathcal{C}_1$ has an infinite computation satisfying $\varphi$, we first construct the tree $T_\varphi$ shown in Figure 6.4. Each node of $T_\varphi$ is labelled with a word from $\{0,1\}^*$ and an operator or atomic formula corresponding to an occurrence of a subformula in $\varphi$. The root node has $\epsilon$ as its first label, and for each node labelled $w$, its left or only child is labelled $w0$ and its right child, if it has one, is labelled $w1$. The second label is assigned in the obvious way, taking the outermost connective of the corresponding subformula.*

*The next step is to identify a set of atoms given by $\varphi$, and transitions between them. Each atom $A$ is a subset of $\{0,1\}^*$ that satisfies the following conditions:*[1]

1. *If $w \in A$ and the node in $T_\varphi$ with first label $w$ has second label $\wedge$, then $w0, w1 \in A$.*

[1]In general, there are also conditions for the operators $\neg$, $\vee$ and $\mathsf{R}$, but those are not relevant for this simple example as they do not occur in $\varphi$.

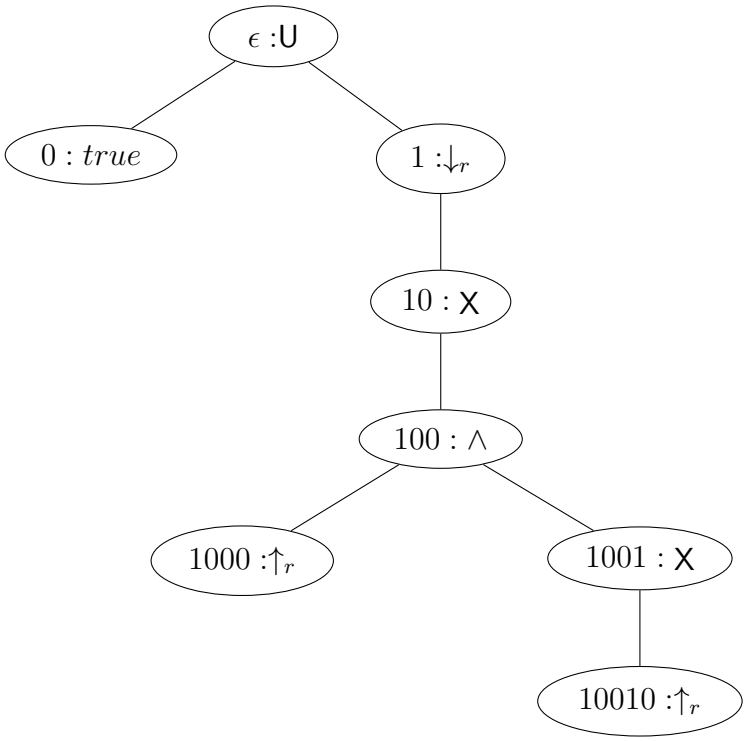


Figure 6.4: The formula tree $T_\varphi$

2. *If $w \in A$ and the node in $T_\varphi$ with first label $w$ has second label* U*, then $\{w0, w1\} \cap A \neq \emptyset$.*

3. *If $w \in A$ and the node in $T_\varphi$ with first label $w$ has second label $\downarrow_r$, then $w0 \in A$.*

*The set of atoms given by $\varphi$ is then $\{\{\epsilon, 0\}, \{\epsilon, 1, 10\}, \{100, 1000, 1001\}, \{10010\}, \emptyset\}$.*

*Next we define the following transition relation between atoms:*

$$\begin{aligned} \{\epsilon, 0\} &\longrightarrow \{\epsilon, 0\} \\ \{\epsilon, 0\} &\longrightarrow \{\epsilon, 1, 10\} \\ \{\epsilon, 1, 10\} &\longrightarrow \{100, 1000, 1001\} \\ \{100, 1000, 1001\} &\longrightarrow \{10010\} \\ \{10010\} &\longrightarrow \emptyset \end{aligned}$$

*The general definition of the transition relation can be found in [DS10]. The intuition is that when all subformulas on the left side hold in some state along a computation, the right side may hold in the next state without violating $\varphi$.*

*The resulting product automaton $\mathcal{C}_1^{(\varphi)}$ is shown in Figure 6.5. It has an initial state $v_0$ and auxiliary states $v_1^{aux}, \ldots, v_6^{aux}$. All other states are of the form $\langle v, A \rangle$ where $v$ is a state in $\mathcal{C}_1$ and $A$ is an atom of $\varphi$. From $v_0$ there is a nondeterministic choice*

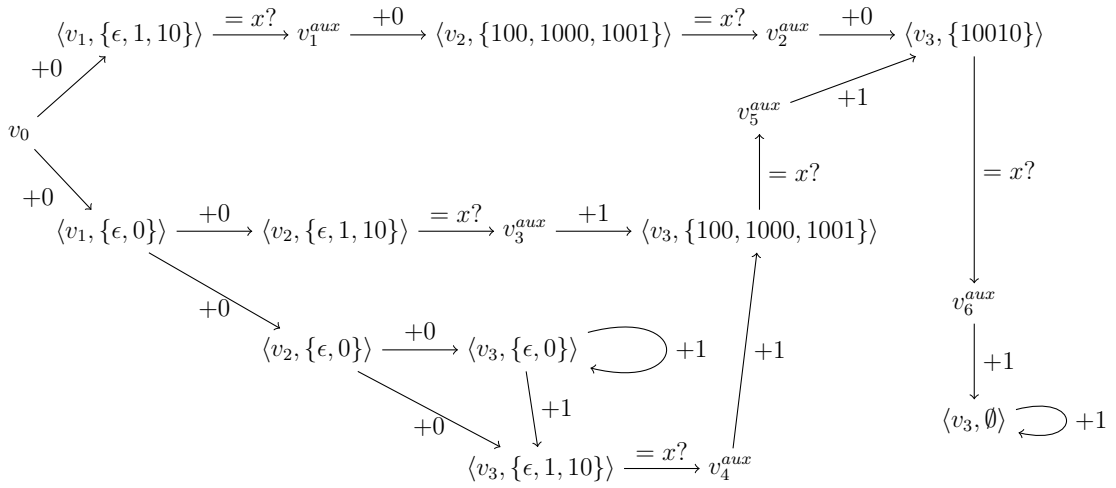


Figure 6.5: The product automaton $\mathcal{C}_1^{(\varphi)}$ of $\mathcal{C}_1$ and $\varphi$.

*between the two atoms including $\epsilon$. The choice of $\{\epsilon, 1, 10\}$ means that $\downarrow_r \mathsf{X}(\uparrow_r \wedge \mathsf{X} \uparrow_r)$ (the right side of $\varphi$) holds in the initial state $v_1$ of $\mathcal{C}_1$, and the choice of $\{\epsilon, 0\}$ means that this subformula will only hold at some point in the future, i.e., in $v_2$ or $v_3$. In any sequence of two edges*

$$\langle v_i, A\rangle \longrightarrow v_k^{aux} \longrightarrow \langle v_j, A'\rangle$$

*the label on the second edge is equal to the label on the edge $(v_i, v_j)$, and the label on the first edge is a test on the counter value. There is one set of final states, $F = \{\langle v_3, \emptyset\rangle\}$.*[2]

*Let us first follow the path from $\langle v_1, \{\epsilon, 1, 10\}\rangle$. The transition to the auxiliary state $v_1^{aux}$ tests whether the current counter value, that is $c$, is equal to the variable $x$. This equality test corresponds to node 1 in $T_\varphi$. Next, $\mathcal{C}_1^{(\varphi)}$ simulates a transition from $v_1$ to $v_2$ in $\mathcal{C}_1$. The transitions to auxiliary states $v_2^{aux}$ and $v_6^{aux}$ include further tests for equality with $x$, which correspond to nodes 1000 and 10010 in $T_\varphi$, respectively. Clearly if we set $x = c$, there is a valid computation starting from $(v_0, c)$ over this path which ends up visiting the final state $\langle v_3, \emptyset\rangle$ infinitely many times.*

*The paths starting at $\langle v_1, \{\epsilon, 0\}\rangle$ all correspond to cases where $\downarrow_r \mathsf{X}(\uparrow_r \wedge \mathsf{X} \uparrow_r)$ does not hold in $v_1$, but only at a later time in $v_2$ or $v_3$. It is easy to see that there is no value for $x$ that allows reachability of the final state along any of these paths.*

*Recall the 1-CA $\mathcal{C}_2$, $\mathcal{C}_3$ and $\mathcal{C}_4$ from Figure 6.3. In the 1-CA $\mathcal{C}_2^{(\varphi)}$, which is constructed in the same way as $\mathcal{C}_1^{(\varphi)}$ but with transition labels from $\mathcal{C}_2$ rather than $\mathcal{C}_1$, there is a valid computation starting in $(v_0, c)$ over the path that goes through $v_3^{aux}$ and $v_5^{aux}$ and ends up visiting $\langle v_3, \emptyset\rangle$ infinitely many times. This means that the right side*

[2]In general, there is a set of final states for each $\mathsf{U}$ operator in the formula.

*of $\varphi$ becomes true in $\mathcal{C}_2$ after one step. Similarly, in $\mathcal{C}_3^{(\varphi)}$, there is a valid computation over any path that goes through $v_4^{aux}$ and $v_5^{aux}$ to $\langle v_3, \emptyset \rangle$, which means that the right side of $\varphi$ becomes true in $\mathcal{C}_3$ after two or more steps. Finally, in $\mathcal{C}_4^{(\varphi)}$, the state $\langle v_3, \emptyset \rangle$ can never be reached, since there is no computation from $(v_1, c)$ in $\mathcal{C}_4$ that satisfies $\varphi$.*

*If $\varphi$ included any negated atomic formulas of the form $\downarrow_{r'} \varphi'$, auxiliary states with disequality tests would be needed in the product automaton. For a general description of how to construct the product automaton and a proof, see [DS10].*

Note that the definition of 1-CA with parameterised tests in [DS10] includes parameterised equality and disequality tests (as in this chapter) together with parameterised inequality tests, i.e., testing whether the counter value is less than or greater than the value of a parameter. However, it is clear from the details of the reduction that only equality and disequality tests are needed, and thus we do not consider inequality tests in this thesis.

## 6.3 Normal Form for Paths

In this section, we show that any valid finite computation of a 1-CA $\mathcal{C} = (V, E, \lambda, \tau)$ can be rewritten to a normal form whose shape only depends on the automaton. Informally, any such computation can be described as a sequence of "take this transition" and "take this cycle $k$ times". We show that the maximum length of a description of this kind is independent of the original computation.

First we show that without loss of generality, any computation can be broken down into a small number of segments that do not contain any transitions with equality tests. The idea is that any segment between two identical equality tests can be omitted.

**Lemma 41.** *Let $\pi$ be a valid finite computation from $(v, c)$ to $(v', c')$. Then there exists a path $\gamma$ such that $\gamma(c)$ is a valid computation from $(v, c)$ to $(v', c')$ and $\gamma$ is of the form $\gamma = \gamma_0 e_1 \gamma_1 e_2 \cdots e_n \gamma_n$, where $e_i$ is an edge with an equality test, $\gamma_i$ is a path without equality tests and $n \leq |E|$.*

*Proof.* Let $\pi'$ be the shortest valid computation from $(v, c)$ to $(v', c')$. We can decompose it as

$$\pi' = (v, c) \xrightarrow{\gamma_0} (v_1, c_1) \xrightarrow{e_1} (v_1', c_1) \xrightarrow{\gamma_1} (v_2, c_2) \cdots (v_n', c_n) \xrightarrow{\gamma_n} (v', c')$$

where for every $i$, $\gamma_i$ is a path without any equality tests and $e_i = (v_i, \mathrm{eq}(c_i), v_i') \in E$ is an equality test. Then clearly $\pi' = \gamma(c)$ where

$$\gamma = \gamma_0 e_1 \gamma_1 e_2 \cdots e_n \gamma_n.$$

Assume for a contradiction that $n > |E|$. Then by the pigeonhole principle, there exists $i < j$ such that $e_i = e_j$. But since $\pi'$ is a valid computation, the two transitions $(v_i, c_i) \xrightarrow{\text{eq}(c_i)} (v'_i, c_i)$ and $(v_j, c_j) \xrightarrow{\text{eq}(c_i)} (v'_j, c_j)$ are the same and $(v_i, c_i) = (v_j, c_j)$. Thus we can delete part of the computation and define

$$\gamma' = \gamma_0 e_1 \gamma_1 \cdots e_i \gamma_j e_{j+1} \cdots e_n \gamma_n.$$

Then $\gamma'(c)$ is a valid computation from $(v, c)$ to $(v', c')$ and is shorter than $\pi'$, which is a contradiction. □

We need to introduce some terminology to formalise our notion of normal form. Given a vertex $v$, $\text{SC}(v)$ (resp. $\text{SC}^+(v)$, $\text{SC}^-(v)$) denotes the *set of equality-free simple* (resp. *positive simple, negative simple*) *cycles* starting at $v$. The set of all equality-free simple (resp. positive simple, negative simple) cycles from all vertices is SC (resp. $\text{SC}^+$, $\text{SC}^-$). Note that each equality-free simple cycle is counted several times in SC: once for each vertex in the cycle.

The *cycle alphabet*, denoted $C$, consists of symbols of the form $\underline{\omega^k}$ where $\omega \in \text{SC}$ and $k \in \mathbb{N}$. Note that this alphabet is infinite. Also note that $\underline{\omega^k}$ is a single symbol, underlined to indicate the difference from the cycle $\omega^k$, which consists of $|\omega|k$ symbols from $E$. For convenience, $\underline{\omega}$ is a shorthand for $\underline{\omega^1}$. We naturally define the start and end of symbol $\underline{\omega^k}$ by the start of $\omega$: $\text{start}(\underline{\omega^k}) = \text{end}(\underline{\omega^k}) = \text{start}(\omega)$.

A *folded path* $\chi$ is a word on the alphabet $E \cup C$: $\chi = s_1 \cdots s_n$ such that $\text{end}(s_i) = \text{start}(s_{i+1})$ for every $i < n$. We also define the natural unfolding of a folded path as a monoid homomorphism $\text{unfold} : (E \cup C)^* \to E^*$ such that $\text{unfold}(e) = e$ for $e \in E$ and $\text{unfold}(\underline{\omega^k}) = \omega^k$ for $\underline{\omega^k} \in C$. The weight of a folded path is the weight of its unfolding.

From now on, until Theorem 48 at the end of this section, we fix an initial counter value $c \in \mathbb{N}$ and we only consider computations starting at $c$ which do not include equality tests. We refer to a folded path $\chi$ as being valid if $\text{unfold}(\chi)(c)$ is a valid computation.

Define the following nondeterministic rewriting system on folded paths. Each rule of the system has a name, a pattern to match against, a condition which must be satisfied for the rule to apply and the result of the rule. We denote by $\chi \rightsquigarrow \chi'$ the fact that $\chi$ rewrites to $\chi'$.

| Rule | Pattern | Result | Condition |
|---|---|---|---|
| `fold` | $\psi\omega\phi$ | $\psi\underline{\omega}\phi$ | $\omega$ is a simple cycle of nonzero weight. |
| `simplify` | $\psi\rho\phi$ | $\psi\phi$ | Nonempty $\rho$, weight(unfold($\rho$)) $= 0$ and end($\psi$) = start($\phi$). |
| `gather`$^+$ | $\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi$ | $\psi\underline{\omega^{k+1}}\rho\underline{\omega^{\ell-1}}\phi$ | Result is valid, $\omega$ is a positive simple cycle and $\ell > 0$. |
| `gather`$^-$ | $\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi$ | $\psi\underline{\omega^{k-1}}\rho\underline{\omega^{\ell+1}}\phi$ | Result is valid, $\omega$ is a negative simple cycle and $k > 0$. |

**Lemma 42.** *If $\chi$ is valid and rewrites to $\chi'$ then $\chi'$ is valid. Furthermore, $\chi$ and $\chi'$ start and end at the same vertex and* weight(unfold($\chi$)) = weight(unfold($\chi'$)).

*Proof.* This is easily checked for each rule:

- `fold`: Clearly unfold($\chi$) = unfold($\chi'$).
- `simplify`: First note that the result is well-formed because of the condition on start and end. The unfolding of the first part ($\psi$) of the path is unchanged, so it remains valid and with the same starting vertex. Since the second part of the path ($\rho$) has weight 0, the counter value is the same at the beginning and end of $\rho$, so the unfolding of the third part ($\phi$) stays the same, and thus valid with the same end vertex. The weight of the unfolded path remains unchanged as the removed part $\rho$ has weight 0.
- `gather`$^\pm$: The condition ensures the result is valid. The start and end vertex clearly do not change, and neither does the weight, since unfold($\chi'$) contains the same edges as unfold($\chi$), only in a different order.

□

**Lemma 43.** *There are no infinite chains of rewriting.*

*Proof.* First we give an informal explanation. The first thing to notice is that the length of a folded path (over alphabet $E \cup C$) never increases after a rewriting operation. The second thing is that the length of a folded path over $E$ (i.e., ignoring symbols from $C$) never increases either. Since rule `simplify` decreases the length, it can only be applied finitely many times. Similarly, rule `fold` decreases the length over $E$ because it replaces a symbol from $E$ by one from $C$. Rules `gather`$^\pm$ are more difficult to analyse because they only reorder the path by replacing symbols from $C$. But as it can be seen, a symbol $\underline{\omega}$, where $\omega$ is a positive cycle, can only move left, and similarly a negative cycle can only move right. Intuitively, this process must be finite because once a positive (negative) cycle reaches the leftmost (rightmost) position, it cannot move anymore.

Formally, we will define a valuation over folded paths and show that it decreases after each application of a rule. First, for any folded path $\chi$ and any given simple cycle $\omega$, define the *$\omega$-projection* $p_\omega(\chi)$ of $\chi$ as the subword only consisting of symbols of the form $\underline{\omega^k}$:

$$p_\omega(e\chi) = p_\omega(\chi) \text{ if } e \in E \qquad p_\omega(\underline{\omega^k}\chi) = \underline{\omega^k} p_\omega(\chi) \qquad p_\omega(\underline{\theta^k}\chi) = p_\omega(\chi) \text{ if } \theta \neq \omega.$$

For any folded path $\chi$, define:

$$(\!|\chi|\!) = (|\chi|, |\chi|_E, \sigma(\chi)) \quad \text{where} \quad \sigma(\chi) = \sum_{\omega \in \mathrm{SC}} \sigma_\omega(p_\omega(\chi))$$

where $|\chi|$ is the word length of $\chi$ (over alphabet $E \cup C$), $|\chi|_E$ is the word length of $\chi$ only counting symbols in $E$ and $\sigma_\omega(p_\omega(\chi))$ is defined as follows:

$$\sigma_\omega\left(\underline{\omega^{k_1}\omega^{k_2}}\cdots\underline{\omega^{k_n}}\right) = \begin{cases} \sum_{i=1}^n ik_i & \text{if weight}(\omega) > 0 \\ 0 & \text{if weight}(\omega) = 0 \\ \sum_{i=1}^n (n+1-i)k_i & \text{if weight}(\omega) < 0. \end{cases}$$

We will now show that $(\!|\chi|\!)$ decreases in lexicographic order each time a rule is applied. In the case of rule `fold`, if $|\omega| > 2$ then clearly $|\chi|$ decreases because we replace several symbols with just one. If $|\omega| = 1$ then $|\chi|$ stays constant but $|\chi|_E$ decreases by one because we replace one symbol from $E$ by one symbol from $C$. Similarly, rule `simplify` decreases $|\chi|$ because we remove a nonzero-length subpath. Since rules `gather`$^+$ and `gather`$^-$ are symmetric, we only prove it for `gather`$^+$. Note that the rule does not change $|\chi|$ or $|\chi|_E$ because it only replaces symbols from $C$ with different symbols from $C$, so we are only concerned with $\sigma(\chi)$.

Assume the rule rewrites $\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi$ into $\psi\underline{\omega^{k+1}}\rho\underline{\omega^{\ell-1}}\phi$. First note that if $\theta \neq \omega$ is a simple cycle, then the $\theta$-projection is the same before and after the rule because the rule does not replace any symbols of the form $\underline{\theta^k}$, so $\sigma_\theta$ does not change. The case of $\sigma_\omega$ is slightly more involved and we need to introduce some notations:

$$p_\omega(\psi) = \underline{\omega^{u_1}}\cdots\underline{\omega^{u_n}}, \qquad p_\omega(\rho) = \underline{\omega^{u_{n+2}}}\cdots\underline{\omega^{u_m}}, \qquad p_\omega(\phi) = \underline{\omega^{u_{m+2}}}\cdots\underline{\omega^{u_q}}$$

and

$$u_{n+1} = k, \quad u_{m+1} = \ell, \quad u'_{n+1} = k+1, \quad u'_{m+1} = \ell - 1, \quad u'_i = u_i \text{ if } i \neq n+1, m+1.$$

Then we can observe that:

$$\sigma_\omega\left(p_\omega\left(\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi\right)\right) = \sigma_\omega\left(\underline{\omega^{u_1}}\cdots\underline{\omega^{u_q}}\right) = \sum_{i=1}^q iu_i, \tag{6.2}$$

$$\sigma_\omega\left(p_\omega\left(\psi\underline{\omega^{k+1}}\rho\underline{\omega^{\ell-1}}\phi\right)\right) = \sigma_\omega\left(\underline{\omega^{u'_1}}\cdots\underline{\omega^{u'_q}}\right) = \sum_{i=1}^q iu'_i. \tag{6.3}$$

Thus:

$$\begin{aligned}(6.2) - (6.3) &= \sum_{i=1}^{q} i(u_i - u'_i) \\ &= (n+1)(u_{n+1} - u'_{n+1}) + (m+1)(u_{m+1} - u'_{m+1}) \\ &= -(n+1) + (m+1) \\ &> 0 \text{ because } m > n.\end{aligned}$$

Thus $\sigma_\omega(\chi)$ decreases after the rule is applied and thus $\sigma(\chi)$ also decreases. □

**Lemma 44.** *If $\psi\rho\phi$ is such that $\rho \in E^*$ and no rule applies, then $|\rho| < |V|$.*

*Proof.* Assume the contrary: if $\rho$ only consists of edges and has length $\geq |V|$, then some vertex is repeated in the vertex sequence of $\rho$. Thus $\rho$ contains a cycle and thus a simple cycle. So rule `fold` applies if the cycle has nonzero weight, or rule `simplify` applies if it has weight zero. □

The idea of the next lemma is to show that given a vertex $v$ in a 1-CA $\mathcal{C} = (V, E, \lambda, \tau)$, some counter values prevent reordering of cycles within the folded path. These counter values act as a "barrier" for the `gather` rules and increase the size of the normal form. We call these values *critical* for positive (resp. negative) cycles and denote them by $B^+(v)$ (resp. $B^-(v)$). Formally, $B^+(v)$ contains:

- $\tau(v) - \text{weight}(\omega)$, for every positive (resp. negative) simple cycle $\omega$,
- $\tau(\text{end}(\gamma)) - \text{weight}(\gamma)$ for every prefix[3] $\gamma$ of every positive (resp. negative) simple cycle $\omega$ starting at $v$.

**Lemma 45.** *Let $\omega$ be a positive cycle and assume that rule `gather+` (resp. `gather-`) does not apply on $\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi$ (which we assume is valid and $k, \ell > 0$) for this particular pattern. Then there exists a (potentially empty) prefix $\mu$ of $\rho$ such that* $\text{unfold}(\psi\underline{\omega^k}\mu)(c)$ *has the form $(v, c) \longrightarrow^* (v', c')$ where $c'$ is critical for $v'$ for positive (resp. negative) cycles,* i.e. *$c' \in B^+(v')$ (resp. $c' \in B^-(v')$). Furthermore $B^+(v')$ and $B^-(v')$ only depend on the automaton and*

$$|B^+(v')| \leq |\,\text{SC}^+\,| \sum_{v \in V} |\tau(v)| \quad \textit{and} \quad |B^-(v')| \leq |\,\text{SC}^-\,| \sum_{v \in V} |\tau(v)|.$$

[3]Other than $\omega$ and the empty prefix. Indeed the empty prefix is impossible because $c_2 \notin \tau(v_1)$ as $\pi$ is valid. And $\omega$ correspond to the previous case of the definition.

*Proof.* We first show the result for positive cycles. Let $\pi = \text{unfold}(\psi\underline{\omega^k}\rho\underline{\omega^\ell}\phi)(c)$ and $\pi' = \text{unfold}(\psi\underline{\omega^{k+1}}\rho\underline{\omega^{\ell-1}}\phi)(c)$. To make things slightly easier to understand, note that:

$$\pi = [\text{unfold}(\psi)\omega^k \,\text{unfold}(\rho)\omega\omega^{\ell-1} \,\text{unfold}(\phi)](c)$$
$$\pi' = [\text{unfold}(\psi)\omega^k\omega \,\text{unfold}(\rho)\omega^{\ell-1} \,\text{unfold}(\phi)](c).$$

Since $\text{unfold}(\rho)\omega$ and $\omega\,\text{unfold}(\rho)$ have the same weight, it is clear that the first ($\text{unfold}(\psi)\omega^k$) and last ($\omega^{\ell-1}\,\text{unfold}(\phi)$) parts of the computation are the same in $\pi$ and $\pi'$, i.e., they have the same counter values. Consequently, if they are valid in $\pi$, the same parts are also valid in $\pi'$. Since by the hypothesis `gather+` does not apply, $\pi'$ is invalid. So there must be an obstruction $(u, d)$ in the middle part ($\omega\,\text{unfold}(\rho)$) of $\pi'$. There are two possibilities.

The first case is when the obstruction $(u, d)$ is in the $\text{unfold}(\rho)$ part of $\pi'$. Note that $d = c^* + \text{weight}(\omega)$, where $(u, c^*)$ is the corresponding configuration in the $\text{unfold}(\rho)$ part of $\pi$. Since $\omega$ is a positive cycle, $d > c^*$ cannot be negative (since $(u, c^*)$ occurs in $\pi$, which is valid). Since we assumed that all computations are free of equality tests, the obstruction must be because of a disequality, i.e., it must be that $d = c^* + \text{weight}(\omega) \in \tau(u)$. Thus $c^* \in \tau(u) - \text{weight}(\omega)$ and $c^*$ is critical for $u$. Then there exists a prefix $\mu$ of $\rho$ such that $\text{unfold}(\psi\underline{\omega^k}\mu)(c) = (v, c) \longrightarrow^* (u, c^*)$ and this shows the result.

The second case is when $(u, d)$ is in the $\omega$ part of the middle part ($\omega\,\text{unfold}(\rho)$) of $\pi'$. Again, it is impossible that the counter value $d$ be negative. Indeed, remember that $\omega$ is a positive cycle and $k > 0$, thus

$$\begin{aligned}\pi' &= [\text{unfold}(\psi)\omega^{k+1} \,\text{unfold}(\rho)\omega^{\ell-1} \,\text{unfold}(\phi)](c)\\ &= [\text{unfold}(\psi)\omega^{k-1}\omega\omega \,\text{unfold}(\rho)\omega^{\ell-1} \,\text{unfold}(\phi)](c)\\ &= (v, c) \xrightarrow{\text{unfold}(\psi)\omega^{k-1}}{}^* (v_1, c_1) \xrightarrow{\omega}{}^* (v_1, c_2) \xrightarrow{\omega}{}^* (v_1, c_3) \xrightarrow{\text{unfold}(\rho)\omega^{\ell-1}\,\text{unfold}(\phi)}{}^* (v'', c'').\end{aligned}$$

We already argued that $(v, c) \longrightarrow^* (v_1, c_2)$ is valid, so in particular $(v_1, c_1) \xrightarrow{\omega}{}^* (v_1, c_2)$ is valid. Note that the obstruction is in the second iteration of $\omega$: $(v_1, c_2) \xrightarrow{\omega}{}^* (v_1, c_3)$. Since $\omega$ is a positive cycle, $c_2 > c_1$. Note that initially the cycle $\omega$ was feasible (with the counter not going negative) starting with a lower counter value ($c_1$) so the counter cannot possibly become negative on the second iteration starting with a higher counter value ($c_2$). Thus, again, the obstruction happens because of a disequality. That is, we can write $\omega = \gamma\gamma'$ such that:

$$\pi' = (v, c) \xrightarrow{\text{unfold}(\psi)\omega^k}{}^* (v_1, c_2) \xrightarrow{\gamma}{}^* (u, d) \xrightarrow{\gamma'}{}^* (v_1, c_3) \xrightarrow{\text{unfold}(\rho)\omega^{\ell-1}\,\text{unfold}(\phi)}{}^* (v'', c'')$$

and the obstruction happens because $d \in \tau(u)$. Note however that $d = c_2 + \text{weight}(\gamma)$ and thus $c_2 \in \tau(u) - \text{weight}(\gamma)$. In this case, $c_2$ is critical for $v_1$. Choose $\mu$ to be the empty word, so that $\text{unfold}(\psi\underline{\omega^k}\mu)(c) = (v, c) \longrightarrow^* (v_1, c_2)$ to show the result.

Observe that the definition of critical values only depend on the automaton itself. Furthermore, the size of $B^+(v)$ can easily be bounded. Indeed, there are $|\,\mathrm{SC}^+\,|$ positive simple cycles, so in the first case of the definition, there are at most $|\,\mathrm{SC}^+\,||\tau(v)|$ values. In the second case, since the cycle $\omega$ is simple, each prefix $\gamma$ of $\omega$ ends at a different vertex. Thus each vertex is visited at most once, and $v$ is not visited because the prefix is not empty or equal to $\omega$. So the second case includes at most an additional $|\,\mathrm{SC}^+\,|\sum_{u\neq v}|\tau(u)|$ values. Finally the total bound is $|\,\mathrm{SC}^+\,|\sum_{v\in V}|\tau(v)|$.

The proof is exactly the same in the negative case except for one detail. This time we move negative cycles to the right so that the middle part of $\pi'$ $(\text{unfold}(\rho)\omega)$ can only get higher counter values than the middle part of $\pi$ $(\omega\,\text{unfold}(\rho))$, as in the positive case. □

**Lemma 46.** *Let $\chi$ be a folded path such that no rule applies on $\chi$. Let $Y = \mathrm{SC}^+$ or $Y = \mathrm{SC}^-$. Then for every $\omega \in Y$, the number of symbols in $\chi$ of the form $\underline{\omega^{\cdot}}$ (the exponent does not matter) is bounded by*

$$|V||Y|\Big(1 + \sum_{v\in V}|\tau(v)|\Big).$$

*Proof.* Without loss of generality, we show the result for $X = \mathrm{SC}^+$. First note that if $\underline{\omega^k}$ appears in $\chi$ and no rule applies, then $k > 0$, otherwise we could apply `simplify` to remove $\underline{\omega^0}$. We can thus decompose the path as:

$$\chi = \phi_0\underline{\omega^{k_1}}\phi_1\underline{\omega^{k_2}}\phi_2\cdots\phi_{n-1}\underline{\omega^{k_n}}\phi_n$$

where $k_i > 0$ and $\phi_i$ does not contain any $\underline{\omega^{\cdot}}$ symbol. Since no rule applies, by Lemma 45, there exist prefixes $\mu_1, \mu_2, \ldots, \mu_{n-1}$ of $\phi_1, \phi_2, \ldots, \phi_{n-1}$ respectively, such that for each $i$:

$$(v, c) \xrightarrow{\phi_0\underline{\omega^{k_1}}\phi_1\cdots\phi_{i-1}\underline{\omega^{k_i}}\mu_i}{}^* (v_i, c_i) \quad \text{where} \quad c_i \in B^+(v_i).$$

Assume for a contradiction that there is a repeated configuration among the $(v_i, c_i)$. Then there exists $i < j$ such that $v_i = v_j$ and $c_i = c_j$. Let $\phi_i = \mu_i\rho$ and $\phi_j = \mu_j\rho'$, and observe that:

$$(v, c) \xrightarrow{\phi_0\underline{\omega^{k_1}}\phi_1\cdots\phi_{i-1}\underline{\omega^{k_i}}\mu_i}{}^* (v_i, c_i) \xrightarrow{\rho\underline{\omega^{k_i+1}}\phi_{i+1}\cdots\phi_{j-1}\underline{\omega^{k_j}}\mu_j}{}^* (v_i, c_i) \xrightarrow{\rho'\underline{\omega^{k_j+1}}\phi_{j+1}\cdots\phi_{n-1}\underline{\omega^{k_n}}\phi_n}{}^* (v', c').$$

Thus the subpath $\rho\underline{\omega^{k_{i+1}}}\phi_{i+1}\cdots\phi_{j-1}\underline{\omega^{k_j}}\mu_j$ has weight 0 and rule `simplify` must apply:

$$\chi \quad \rightsquigarrow \quad \phi_0\underline{\omega^{k_1}}\phi_1\cdots\phi_{i-1}\underline{\omega^{k_i}}\mu_i\rho'\underline{\omega^{k_{j+1}}}\phi_{j+1}\cdots\phi_{n-1}\underline{\omega^{k_n}}\phi_n$$

which is a contradiction because we assumed that no rule can apply on $\chi$.

Consequently, for any $i \neq j$, we have $(v_i, c_i) \neq (v_j, c_j)$. But remember that $c_i \in B^+(v_i)$, thus $(v_i, c_i) \in A$ where:

$$A = \bigcup_{v\in V} \{v\} \times B^+(v).$$

This shows that $n - 1 \leq |A|$. Indeed, by the pigeonhole principle, some pair $(v_i, c_i)$ would be repeated if $n - 1 > |A|$. We can easily bound the size of $A$ using the bound on $B^+(v)$ from Lemma 45:

$$|A| \leq \sum_{v\in V} |B^+(v)| \leq |V||\operatorname{SC}^+|\sum_{v\in V}|\tau(v)|.$$

Finally we have

$$n \leq |V||\operatorname{SC}^+|\sum_{v\in V}|\tau(v)| + 1 \leq |V||\operatorname{SC}^+|\Big(1 + \sum_{v\in V}|\tau(v)|\Big)$$

because $|V| \geq 1$ and $|\operatorname{SC}^+| \geq 1$ unless there are no positive cycles, in which case $n = 0$ anyway. □

**Lemma 47.** *Let $\pi$ be a valid finite computation (without equality tests) from $(v, c)$ to $(v', c')$. Then there exists a folded path $\chi$ such that* $\operatorname{unfold}(\chi)(c)$ *is a valid computation from $(v, c)$ to $(v', c')$, the length of* $\operatorname{unfold}(\chi)(c)$ *is at most that of $\pi$ and the word length of $\chi$ is bounded by:*

$$|V| + |V|^2|\operatorname{SC}|^2\Big(1 + \sum_{v\in V}|\tau(v)|\Big)$$

*Proof.* Let $\chi_0$ be the path defined by $\pi$: it is a word over alphabet $E$ and is thus a (trivial) folded path. By definition $\operatorname{unfold}(\chi_0(c)) = \pi$ is a valid computation from $(v, c)$ to $(v', c')$ and the length of $\operatorname{unfold}(\chi_0(c))$ is equal to that of $\pi$. Let $\chi$ be any rewriting of $\chi_0$ such that no rule applies on $\chi$: it exists because there are no infinite rewriting chains by Lemma 43. By Lemma 42, $\operatorname{unfold}(\chi(c))$ is still a valid computation from $(v, c)$ to $(v', c')$. Let $\omega$ be a simple cycle: note that it is either positive or negative, because rule `simplify` removes zero-weight cycles. Then by Lemma 46, the number of symbols of the form $\underline{\omega^{\cdot}}$ appearing in $\chi$ is bounded by[4]:

$$|V||\operatorname{SC}|\Big(1 + \sum_{v\in V}|\tau(v)|\Big) \tag{6.4}$$

[4]Since obviously $\max(|\operatorname{SC}^+|, |\operatorname{SC}^-|) \leq |\operatorname{SC}|$.

and thus the total number of symbols in $\chi$ of the form $\underline{\omega^{\cdot}}$ for any $\omega$ is bounded by:

$$|V||\,\mathrm{SC}\,|^2\Big(1+\sum_{v\in V}|\tau(v)|\Big). \tag{6.5}$$

Furthermore, inbetween symbols of the form $\underline{\omega^{\cdot}}$, there can be subpaths consisting of symbols in $E$ only, so $\chi$ is of the form

$$\chi = \phi_0\underline{\omega_1^{k_1}}\phi_1\underline{\omega_2^{k_2}}\cdots\underline{\omega_n^{k_n}}\phi_n$$

where $\phi_i \in E^*$ and $\omega_i \in \mathrm{SC}$ for all $i$. By the reasoning above, $n \leq (6.5)$. Furthermore, by Lemma 44, $\phi_i < |V|$ for all $i$. It follows that the total length of $\chi$ is bounded by

$$\begin{aligned}(n+1)(|V|-1)+n &\leq |V|+n|V| \\ &\leq |V|+|V|^2|\,\mathrm{SC}\,|^2\Big(1+\sum_{v\in V}|\tau(v)|\Big).\end{aligned}$$

Finally the length of $\mathrm{unfold}(\chi(c))$ at most that of $\pi$ because the rewriting system does not increase the length of the path and the length of $\mathrm{unfold}(\chi_0(c))$ is equal to that of $\pi$. □

The main result of this section shows that any valid computation has an equivalent valid computation given by a folded path whose length only depends on the automaton.

**Theorem 48.** *Let $\pi$ be a valid finite computation from $(v,c)$ to $(v',c')$. Then there exists a folded path $\chi$ such that $\chi(c)$ is a valid computation from $(v,c)$ to $(v',c')$, the length of* $\mathrm{unfold}(\chi(c))$ *is at most that of $\pi$ and the word length of $\chi$ is bounded by:*

$$|E|\left(1+|V|+|V|^2|\,\mathrm{SC}\,|^2\Big(1+\sum_{v\in V}|\tau(v)|\Big)\right).$$

*Proof.* Apply Lemma 41 to isolate the equality tests (at most $|E|$ of them) and apply Lemma 47 to each equality-free subcomputation. We can improve the bound slightly by noticing that there can only be up to $|E|$ equality-free subcomputations (and not $|E|+1$). Indeed, if there are $|E|$ different equality tests in the path, there are no further edges available for equality-free computations, and the word length is at most $|E|$. □

## 6.4 Reachability with Parameterised Tests

In this section we will show that both the reachability problem and the generalised repeated control-state reachability problem for 1-CA with parameterised tests are decidable, via a symbolic encoding of folded paths, making use of the normal form from the previous section. The result of this encoding is a formula of Presburger arithmetic.

Recall that $C = \{\underline{\omega^k} : \omega \in \mathrm{SC}, k \in \mathbb{N}\}$. Let $C' = \{\underline{\omega^\cdot} : \omega \in \mathrm{SC}\}$. We define a *path shape* to be a word over the alphabet $E \cup C'$: $\xi = t_1 \dots t_n$ such that $\mathrm{end}(t_i) = \mathrm{start}(t_{i+1})$, where $\mathrm{start}(\underline{\omega^\cdot}) = \mathrm{end}(\underline{\omega^\cdot}) = \mathrm{start}(\omega)$. Given a path shape $\xi = \gamma_0 \underline{\omega_1^\cdot} \gamma_1 \dots \underline{\omega_n^\cdot} \gamma_n$ with $\gamma_i \in E^*$, we write $\xi(k_1, \dots, k_n)$ for the folded path $\gamma_0 \underline{\omega_1^{k_1}} \gamma_1 \dots \underline{\omega_n^{k_n}} \gamma_n$. The advantage of working with path shapes rather than folded paths is that the former are words over a finite alphabet.

**Lemma 49.** *Given a 1-CA $\mathcal{C} = (V, E, X, \lambda, \tau)$ with parameterised tests and configurations $(v, c)$ and $(v', c')$, and given a path shape $\xi = t_1 t_2 \dots t_n \in (E \cup C')^*$, there exists a Presburger arithmetic formula $\varphi_{comp}^{(\xi),(v,c),(v',c')}(\boldsymbol{k}, \boldsymbol{x})$, with free variables $\boldsymbol{x}$ corresponding to the parameters $X$ and $\boldsymbol{k}$ corresponding to exponents to be substituted in $\xi$, which evaluates to true if and only if* $\mathrm{unfold}(\xi(\boldsymbol{k}))(c)$ *is a valid computation from $(v, c)$ to $(v', c')$.*

*Proof.* Assume first that $\xi$ does not include any equality tests. We define a formula $\varphi_{valid,noeq}^{(t)}(\boldsymbol{k}, \boldsymbol{x}, y)$ which, given an equality-free symbol $t \in E \cup C'$ and an integer $y$, evaluates to true if and only if $\mathrm{unfold}(t(\boldsymbol{k}))(y)$ is a valid computation. There are two cases:

- $t \in E$. Then $\varphi_{valid,noeq}^{(t)}(\boldsymbol{x}, y) \equiv y \geq 0 \wedge y + \mathrm{weight}(t) \geq 0 \wedge y \notin \tau(\mathrm{start}(t))$.
- $t \in C'$, that is, $t(\boldsymbol{k}) = \underline{\omega^k}$ for some simple cycle $\omega = e_1 e_2 \dots e_\ell$ and $k \in \boldsymbol{k}$. Then

$$\begin{aligned}\varphi_{valid,noeq}^{(t)}(\boldsymbol{k}, \boldsymbol{x}, y) \equiv y + k\,\mathrm{weight}(\omega) \geq 0 \wedge \forall k' \,(0 \leq k' < k) \Rightarrow \\ \bigwedge_{i=1}^{\ell} \left( y + k'\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) \geq 0 \wedge \right. \\ \left. y + k'\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) \notin \tau(\mathrm{start}(e_i)) \right)\end{aligned}$$

Note that for each edge $e \in E$, $\mathrm{weight}(e)$ is a constant, given by the automaton, and $\mathrm{weight}(\omega)$ is a shorthand for $\sum_{i=1}^{\ell} \mathrm{weight}(e_i)$, which is also a constant. So the only type of multiplication in the formula is by a constant. A formula of the form $a \notin \tau(u)$ is a shorthand for $\bigwedge_{b \in \tau(u)} a \neq b$, which is clearly a Presburger arithmetic formula.

Since $\mathcal{C}$ has parameterised tests, in general some of these disequalities include variables from $\boldsymbol{x}$. We can now define a formula with the required property in the case where $\xi$ does not include any equality tests:

$$\varphi_{comp,noeq}^{(\xi),(v,c),(v',c')}(\boldsymbol{k},\boldsymbol{x}) \equiv \left(\bigwedge_{i=1}^{n-1} \mathrm{end}(t_i) = \mathrm{start}(t_{i+1})\right) \wedge \mathrm{start}(t_1) = v \wedge \mathrm{end}(t_n) = v' \wedge$$
$$\sum_{i=1}^{n} \mathrm{weight}(t_i(\boldsymbol{k})) = c' - c \wedge \bigwedge_{i=1}^{n} \varphi_{valid,noeq}^{(t_i)}(\boldsymbol{k},\boldsymbol{x}, c + \sum_{j=1}^{i-1} \mathrm{weight}(t_j(\boldsymbol{k}))),$$

where we use the shorthand $\mathrm{weight}(s)$ for $s \in E \cup C$: if $s \in E$ then $\mathrm{weight}(s)$ is a constant as above, and if $s \in C$ then it is of the form $\underline{\omega^k}$ and $\mathrm{weight}(s) = k \sum_{e \in \omega} \mathrm{weight}(e)$. Again, the only multiplications are by constants, so the resulting formula is a formula of Presburger arithmetic.

Finally, in the case where $\xi$ includes equality tests, we split $\mathrm{unfold}(\xi)$ at the $t_i$ which are equality tests, and construct a formula $\varphi_{comp,noeq}$ as above for each equality-free part of $\xi$. $\varphi_{comp}^{(\xi),(v,c),(v',c')}(\boldsymbol{k},\boldsymbol{x})$ is the conjunction of these formulas. □

**Remark 50.** *For simplicity, we have used a universal quantifier in $\varphi_{valid,noeq}^{(t)}(\boldsymbol{k},\boldsymbol{x},y)$ to express that $k$ iterations of a cycle yield a valid computation. In fact it is possible to rewrite $\varphi_{valid,noeq}^{(t)}(\boldsymbol{k},\boldsymbol{x},y)$ as a purely existential formula, with a polynomial blowup. Let $\omega = e_1 \cdots e_\ell$ be a cycle and suppose we want to check that $\omega^k(y)$ is a valid computation. Let $u = \mathrm{start}(e_i)$ be a state on the cycle. First we need to express that the counter value at $u$ is never negative along $\omega^k(y)$. Since the counter value at $u$ is monotone during the $k$ iterations of the cycle (it increases if $\omega$ is positive and decreases if $\omega$ is negative), we only need check that it is nonnegative at the first and last iteration:*

$$y + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) \geq 0 \wedge y + (k-1)\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) \geq 0.$$

*Next, for each $b \in \tau(u)$, we need to check that the cycle avoids $b$ in $u$. Without loss of generality, assume that $\omega$ is positive. Then the counter value at $u$ increases after each iteration. We can now perform a case analysis on the three ways to satisfy a disequality test during the $k$ iterations of $\omega$:*

- *The value at the first iteration is already bigger than $b$:*

$$y + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) > b.$$

- *The value at the last iteration is less than $b$:*

$$y + (k-1)\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) < b.$$

- *There is an iteration $k'$, with $0 \leq k' < k-1$, at which the counter value is less than $b$, but where at the next iteration $k'+1$ the counter value is bigger than $b$:*

$$\exists k' \, (0 \leq k' < k-1) \wedge y + k'\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) < b$$
$$\wedge\, y + (k'+1)\,\mathrm{weight}(\omega) + \sum_{j=1}^{i-1} \mathrm{weight}(e_j) > b.$$

*Finally, we can use a conjunction over all vertices in $\omega$ to get a formula which is equivalent to $\varphi^{(t)}_{valid,noeq}(\boldsymbol{k}, \boldsymbol{x}, y)$ but has no universal quantifiers.*

**Lemma 51.** *Let $\mathcal{C} = (V, E, X, \lambda, \tau)$ be a 1-CA with parameterised tests, and let $(v, c)$ and $(v', c')$ be given configurations of $\mathcal{C}$. Then there exists a Presburger arithmetic formula $\varphi^{(v,c),(v',c')}_{reach}(\boldsymbol{x})$ which evaluates to true if and only if there is a valid computation from $(v, c)$ to $(v', c')$ in $\mathcal{C}$, as well as a formula $\varphi^{(v,c),(v',c')}_{reach_+}$ which is true if and only if there is such a computation of length at least 1.*

*Proof.* Note that the bounds on the length of computations in 1-CA from the previous section do not depend on the values occurring in equality or disequality tests. That is, if there is a valid computation $(v, c) \xrightarrow{\pi}{}^{*} (v', c')$ for any given values of the parameters, then there is a folded path $\chi$ of word length at most $p(\mathcal{C})$ such that $(v, c) \xrightarrow{\mathrm{unfold}(\chi(c))}{}^{*} (v', c')$ is a valid computation, where $p$ is the polynomial function given in Theorem 48. Equivalently, there is a path shape $\xi$ of word length at most $p(\mathcal{C})$ and there exist values $\boldsymbol{k}$ such that $(v, c) \xrightarrow{\mathrm{unfold}(\xi(\boldsymbol{k})(c))}{}^{*} (v', c')$ is a valid computation.

Since path shapes are words over a finite alphabet, we can express this property as a finite disjunction

$$\varphi^{(v,c),(v',c')}_{reach}(\boldsymbol{x}) \equiv \exists \boldsymbol{k} \bigvee_{|\xi| \leq p(\mathcal{C})} \varphi^{(\xi),(v,c),(v',c')}_{comp}(\boldsymbol{k}, \boldsymbol{x}).$$

For $\varphi_{reach_+}$, we simply change the disjunction to be over all $\xi$ such that $1 \leq |\xi| \leq p(\mathcal{C})$. □

**Lemma 52.** *Let $\mathcal{C} = (V, E, X, \lambda, \tau)$ be a 1-CA with parameterised tests, let $F \subseteq V$ be a set of final states, and let $(v, c)$ be the initial configuration of $\mathcal{C}$. Then there exists a Presburger arithmetic formula $\varphi^{(v,c),(F)}_{\textit{rep-reach}}(\boldsymbol{x})$ which evaluates to true if and only if there is a valid infinite computation $\pi$ which starts in $(v, c)$ and visits at least one state in $F$ infinitely often.*

*Proof.* Suppose there is an infinite computation which starts in $(v, c)$ and visits a state $u \in F$ infinitely often. Equivalently, there is a counter value $d \in \mathbb{N}$ such that $(v, c) \longrightarrow^* (u, d)$ is a valid (finite) computation, and there is a cycle $\omega$ with $\text{start}(\omega) = u$ such that $\omega^k(d)$ is a valid computation for all $k \in \mathbb{N}$. There are two possible cases:

- $\text{weight}(\omega) = 0$, so $\omega^k(d)$ is valid for all $k$ if and only if $\omega(d)$ is valid.
- $\text{weight}(\omega) > 0$, so it might be possible to start from $(u, d)$ and follow the edges of $\omega$ a finite number of times before an obstruction occurs. However, if $\omega$ can be taken an arbitrary number of times, then the counter value will tend towards infinity, so we are free to choose $\omega$ to be an equality-free simple cycle, and $d$ to be high enough to guarantee that if $\omega$ can be taken once without obstructions, it can be taken infinitely many times.

The resulting formula is then

$$\varphi^{(v,c),(F)}_{\textit{rep-reach}}(\boldsymbol{x}) \equiv \exists d \bigvee_{u \in F} \Bigg( \varphi^{(v,c),(u,d)}_{\textit{reach}}(\boldsymbol{x}) \wedge \Big( \varphi^{(u,d),(u,d)}_{\textit{reach}_+}(\boldsymbol{x}) \vee (d > M(\boldsymbol{x}) \wedge \exists d' \bigvee_{\omega \in \text{SC}^+} \varphi^{(\underline{\omega}^{\cdot}),(u,d),(u,d')}_{\textit{comp,noeq}}(1, \boldsymbol{x})) \Big) \Bigg)$$

where $M(\boldsymbol{x}) = \max\left(\bigcup_{v \in V} \tau(v)\right) - \sum\{\text{weight}(e) : e \in E, \text{weight}(e) < 0\}$. The sum over negative edge weights ensures that the counter always stays above $\max\left(\bigcup_{v \in V} \tau(v)\right)$ along the computation $\omega(d)$, since each edge is taken at most once in $\omega$. Since $\omega$ is a positive cycle, this implies that the counter always stays above all bad values along $\omega(d^k)$ for each $k \in \mathbb{N}$, so no obstructions can occur. □

**Theorem 53.** *Both the reachability problem and the generalised repeated control-state reachability problem are decidable for 1-CA with parameterised tests.*

*Proof.* Given a 1-CA $\mathcal{C} = (V, E, X, \lambda, \tau)$ with parameterised tests and configurations $(v, c)$ and $(v', c')$, to check if there exist values for the parameters $X$ such that there is a valid computation from $(v, c)$ to $(v', c')$, we use Lemma 51 to construct the formula $\exists \boldsymbol{x}\, \varphi^{(v,c),(v',c')}_{\textit{reach}}(\boldsymbol{x})$.

To solve the generalised repeated control-state reachability problem for a 1-CA $\mathcal{C} = (V, E, X, \lambda, \tau)$ with sets of final states $F_1, \ldots, F_n \subseteq V$ and initial configuration $(v, c)$, note that this problem can easily be reduced to the simpler case where $n = 1$, using a translation similar to the standard translation from generalised Büchi automata to Büchi automata. In the case where $n = 1$, we can use Lemma 52 to construct the formula $\exists \boldsymbol{x}\, \varphi^{(v,c),(F_1)}_{\textit{rep-reach}}(\boldsymbol{x})$. □

**Corollary 54.** *The existential model checking problem for flat Freeze LTL on 1-CA is decidable.*

# Chapter 7

# Conclusion

We have addressed several open problems related to linear arithmetic and one-counter automata. The decision problem for the existential linear theory of the $p$-adic numbers was known to be **NP**-hard and in **EXPTIME**, and we improved the upper bound to **CH**, that is, the Counting Hierarchy, which is included in **PSPACE**. The complexity of deciding existential sentences of Presburger arithmetic with divisibility was known to be between **NP** and **2NEXPTIME**, and we improved the upper bound to **NEXPTIME**. As for one-counter automata, both the LTL synthesis problem for a fully parameterised version of classical one-counter automata and the model checking problem for flat Freeze LTL on classical one-counter automata (and, as a requirement, the repeated reachability problem on one-counter automata with parameterised equality and disequality tests) have been stated as open problems in the literature; we solved the first of them positively and reduced the second to the decision problem of a first-order theory whose decidability proof is currently under review.

There remain the open problems of determining the precise complexity of these problems. In [Wei88], Weispfenning asks whether the decision problem for the existential linear theory of $\mathbb{Q}_p$ is in **NP**. The obstacle to obtaining such a bound is the need to establish a polynomial bound on the size of the integer constants generated in the process of quantifier elimination. In this respect it is interesting to observe that the quantifier elimination procedure in Section 3.4 is formally very similar to the process of reducing a matrix to echelon form through elementary row operations. Now if one uses Gaussian elimination to reduce a matrix to echelon form then there is a polynomial bound on the size of any matrix entry appearing during the reduction process: in fact all such entries are quotients of minors of the input matrix [GLS93]. Our quantifier elimination procedure in 3.4 is essentially a generalisation of Gaussian elimination, where in every step there may be more than one pivot row. We leave for

future work the question of whether the arguments of [GLS93] can be generalised to our reduction process for the existential linear theory of $\mathbb{Q}_p$.

For satisfiable quantifier-free formulas of Presburger arithmetic with divisiblity, we have established a tight bound on the size of the smallest solution, which is singly exponential in the number of variables of the formula. An intriguing open problem is to find matching complexity bounds for the decision problem of existential Presburger arithmetic with divisibility. This problem has been open since the original proofs of decidability by Lipshitz and Bel'tyukov in 1976. It seems unlikely that our method of proving the **NEXPTIME** upper bound can be directly improved to give a better result, as our proof is by an optimal bound on the smallest solution. While it may seem unlikely that the problem is in **NP** because of the singly exponential lower bound on a satisfying assignment, the possibility of improving the **NEXPTIME** upper bound should not be completely discarded for that reason. Consider for example the existential theory of the reals (including both addition and multiplication), which is in **PSPACE** despite having an exponential lower bound on satisfying assignments of formulas. [Can88]

It is worth noting that existential Presburger arithmetic with divisibility is on the boundary of decidability, in that extensions of this theory very easily become undecidable. The decision problem of existential Presburger arithmetic with multiplication is undecidable, and Lipshitz showed in [Lip77] that the existential linear theory of the ring of integers in any quadratic number field, including divisibility, is undecidable.

Concerning the second part of this thesis, decidability of the LTL synthesis problem for one-counter automata was stated as an open problem in [GHOW10]. The same paper includes a proof that the corresponding synthesis problem for CTL is undecidable. The proof is by reduction from Hilbert's Tenth Problem and relies on the branching structure of CTL. The fact that this problem could be decidable for LTL seems to confirm the general observation from [Var01] that synthesis problems tend to be harder for branching time than for linear time logic. A lower bound of **PSPACE** for the LTL synthesis problem immediately follows from the fact that LTL model checking for Kripke structures is **PSPACE**-complete. An interesting open problem would be to improve this lower bound if decidablity of the LTL synthesis problem can be established.

The basic reachability problems for the types of one-counter automata that we consider (parameterised one-counter automata with zero tests and one-counter automata with parameterised equality and disequality tests) can be seen as special cases of a long-standing open problem identified by Ibarra *et al.* [IJTW93] which asks to

decide reachability on a class of automata with a single integer-valued counter, sign tests, and parameterised updates. The decidability of reachability in the case of a nonnegative integer-valued counter and parameterised equality and inequality tests (and integer updates without parameters) is also open. [DS10]

For reachability in one-counter automata with parameterised equality and disequality tests, we have concentrated on showing decidability rather than achieving optimal complexity. For example, we have reduced the reachability problem to the decision problem for the class of sentences of Presburger arithmetic with quantifier prefix $\exists^*\forall^*$. We explained in Remark 50 that in fact the reduction can be refined to yield a (polynomially larger) purely existential sentence.

Another important determinant of the complexity of our procedure is the dependence of the symbolic encoding of computations (via path shapes) in Section 6.4 on the number of simple cycles in the underlying control graph of the one-counter automaton. The number of such cycles may be exponential in the number of vertices. It remains to be seen whether it is possible to give a more compact symbolic representation, for example in terms of the Parikh image of paths.

As it stands, our procedure for model checking flat Freeze LTL formulas on classical one-counter automata works as follows. From the flat Freeze LTL formula and the automaton, we build a one-counter automaton with parameterised tests (of exponential size). We then guess the normal form of the path shapes (of exponential size in the size of the automaton). We finally check the resulting existential Presburger formula. Since the Presburger formula has size doubly exponential in the size of the input, we get a naive upper bound of **2NEXPTIME** for our algorithm. Improving this bound is a subject of ongoing work.

Another interesting complexity question concerns configuration reachability in one-counter automata with non-parameterised equality and disequality tests. For automata with only equality tests and with counter updates in binary, reachability is known to be **NP**-complete [HKOW09]. If inequality tests are allowed then reachability is **PSPACE**-complete [FJ15]. Now automata with equality and disequality tests are intermediate in expressiveness between these two models and the complexity of reachability in this case is open as far as we know.

# Bibliography


[ABD14] E. Allender, N. Balaji, and S. Datta. Low-depth uniform threshold circuits and the bit-complexity of straight line programs. In *MFCS*, volume 8635 of *LNCS*, pages 13–24. Springer, 2014.

[ABH01] E. Allender, D. A. M. Barrington, and W. Hesse. Uniform circuits for division: Consequences and problems. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity*, pages 150–159. IEEE Computer Society, 2001.

[ABKM09] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.

[Bak11] A. Baker. An introduction to $p$-adic numbers and $p$-adic analysis. *available at http://www.maths.gla.ac.uk/%7Eajb/dvi-ps/padicnotes.pdf*, 2011.

[BBH+06] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro, and T. Vojnar. Programs with lists are counter automata. In *Proceedings of CAV*, volume 4144 of *LNCS*, pages 517–531. Springer, 2006.

[Bel76] A. Bel'tyukov. Decidability of the universal theory of the natural numbers with addition and divisibility (in Russian). *Zapiski Nauchnyh Seminarov LOMI*, 60:15–28, 1976.

[BI05] M. Bozga and R. Iosif. On decidability within the arithmetic of addition and divisibility. In *Proceedings of FoSSaCS*, volume 3441 of *Lecture Notes in Computer Science*, pages 425–439. Springer, 2005.

[BKL+97] P. Berman, M. Karpinski, L. L. Larmore, W. Plandowski, and W. Rytter. On the complexity of pattern matching for highly compressed two-dimensional texts. In *Proceedings of CPM*, volume 1264 of *LNCS*, pages 40–51. Springer, 1997.

[BL12] M. Bojanczyk and S. Lasota. Minimization of semilinear automata. *CoRR*, abs/1210.4980, 2012.

[BMOW08] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. On expressiveness and complexity in real-time model checking. In *Proceedings of ICALP*, volume 5126 of *LNCS*, pages 124–135. Springer, 2008.

[BO14] D. Bundala and J. Ouaknine. Advances in parametric real-time reasoning. In *Proceedings of MFCS*, volume 8634 of *Lecture Notes in Computer Science*, pages 123–134. Springer, 2014.

[BT76] I. Borosh and L. B. Treybig. Bounds on positive integral solutions of linear Diophantine equations. *Proceedings of the American Mathematical Society*, 55(2):299–304, 1976.

[Can88] J. F. Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of STOC*, pages 460–467. ACM, 1988.

[CC00] H. Comon and V. Cortier. Flatness is not a weakness. In *Proceedings of CSL*, volume 1862 of *LNCS*. Springer, 2000.

[Chu36] A. Church. An unsolvable problem of elementary number theory. *American journal of mathematics*, 58:345–363, 1936.

[CLO07] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag New York, 3rd edition, 2007.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[Coh69] P. Cohen. Decision procedures for real and $p$-adic fields. *Communications on Pure and Applied Mathematics*, XXII:131–151, 1969.

[Coh93] H. Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993.

[CR04] C. Chitic and D. Rosu. On validation of XML streams using finite state machines. In *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*, pages 85–90. ACM, 2004.

[DL06] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. In *Proceedings of LICS*, pages 17–26. IEEE Computer Society, 2006.

[DLN05] S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in constraint LTL: decidability and complexity. In *Proceedings of TIME*, pages 113–121, 2005.

[DLS08] S. Demri, R. Lazić, and A. Sangnier. Model checking freeze LTL over one-counter automata. In *Proceedings of FOSSACS*, volume 4962 of *LNCS*, pages 490–504, 2008.

[DMV95] A. Degtyarev, Y. Matiyasevich, and A. Voronkov. Simultaneous rigid $E$-unification is not so simple. Technical Report 104, UPMAIL, 1995.

[DMV96] A. Degtyarev, Y. Matiyasevich, and A. Voronkov. Simultaneous $E$-unification and related algorithmic problems. In *Proceedings of LICS*, pages 494–502. IEEE Computer Society, 1996.

[DS10] S. Demri and A. Sangnier. When model-checking freeze LTL over counter machines becomes decidable. In *Proceedings of FOSSACS*, volume 6014 of *LNCS*, pages 176–190, 2010.

[DV01] A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi. In *Handbook of Automated Reasoning (in 2 volumes)*, pages 611–706. Elsevier and MIT Press, 2001.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proceedings of LICS*, pages 352–359, 1999.

[FH97] X. G. Fang and G. Havas. On the worst-case complexity of integer gaussian elimination. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 28–31. ACM, 1997.

[FJ15] J. Fearnley and M. Jurdzinski. Reachability in two-clock timed automata is PSPACE-complete. *Inf. Comput.*, 243:26–36, 2015.

[FL02] A. Finkel and J. Leroux. How to compose Presburger-accelerations: Applications to broadcast protocols. In *Proceedings of FSTTCS*, pages 145–156. Springer, 2002.

[FR74] M. J. Fischer and M. O. Rabin. Super-exponential complexity of Presburger arithmetic. In *Proceedings of SIAM-AMS*, pages 27–41, 1974.

[Fre03] T. French. Quantified propositional temporal logic with repeating states. In *Proceedings of TIME-ICTL*, pages 155–165. IEEE Computer Society, 2003.

[GHOW10] S. Göller, C. Haase, J. Ouaknine, and J. Worrell. Model checking succinct and parametric one-counter automata. In *Proceedings of ICALP*, volume 6199 of *Lecture Notes in Computer Science*, pages 575–586. Springer, 2010.

[GI82] E. M. Gurari and O. H. Ibarra. Two-way counter machines and Diophantine equations. *J. ACM*, 29(3):863–873, 1982.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1993.

[Haa12] C. Haase. *On the Complexity of Model Checking Counter Automata*. Ph.D. Thesis, University of Oxford, 2012.

[Hen97] K. Hensel. über eine neue Begründung der Theorie der algebraischen Zahlen. In *Jahresbericht der Deutschen Mathematiker-Vereinigung*, pages 83–88, 1897.

[HKOW09] C. Haase, S. Kreutzer, J. Ouaknine, and J. Worrell. Reachability in succinct and parametric one-counter automata. In *Proceedings of CONCUR*, volume 5710 of *Lecture Notes in Computer Science*, pages 369–383. Springer, 2009.

[Iba79] O. H. Ibarra. Restricted one-counter machines with undecidable universe problems. *Mathematical Systems Theory*, 13:181–186, 1979.

[ID04] O. H. Ibarra and Z. Dang. On two-way finite automata with monotonic counters and quadratic Diophantine equations. *Theor. Comput. Sci.*, 312(2-3):359–378, 2004.

[ID06] O. H. Ibarra and Z. Dang. On the solvability of a class of Diophantine equations and applications. *Theor. Comput. Sci.*, 352(1):342–346, 2006.

[IJTW93] O. H. Ibarra, T. Jiang, N. Trân, and H. Wang. New decidability results concerning two-way counter machines and applications. In *ICALP*, volume 700 of *LNCS*. Springer, 1993.

[ISD+02] O. H. Ibarra, J. Su, Z. Dang, T. Bultan, and R. A. Kemmerer. Counter machines and verification problems. *Theor. Comput. Sci.*, 289(1):165–189, 2002.

[Lec15] A. Lechner. Synthesis problems for one-counter automata. In *Proceedings of RP*, 2015.

[Lip76] L. Lipshitz. The Diophantine problem for addition and divisibility. *Transactions of the American Mathematical Society*, 235:271–283, 1976.

[Lip77] L. Lipshitz. Undecidable existential problems for addition and divisibility in algebraic number rings. In *Proceedings of the American Mathematical Society*, volume 64, pages 122–128, 1977.

[Lip81] L. Lipshitz. Some remarks on the Diophantine problem for addition and divisibility. *Bull. Soc. Math. Belg. Sér. B*, 33(1):41–52, 1981.

[LLT04] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *Research Report LSV-04-16*. LSV, ENS de Cachan, 2004.

[LMO+16] A. Lechner, R. Mayr, J. Ouaknine, A. Pouly, and J. Worrell. Model checking flat freeze LTL on one-counter automata. In *Proceedings of CONCUR*, 2016.

[LOW15] A. Lechner, J. Ouaknine, and J. Worrell. On the complexity of linear arithmetic with divisibility. In *Proceedings of LICS*, 2015.

[LP05] A. Lisitsa and I. Potapov. Temporal logic with predicate lambda-abstraction. In *Proceedings of TIME*, pages 147–155. IEEE Computer Society, 2005.

[Mah58] K. Mahler. On the Chinese remainder theorem. *Math. Nach.*, 18:120–122, 1958.

[Mat70] Y. Matiyasevich. Enumerable sets are Diophantine. *Journal of Soviet Mathematics*, 11:354–358, 1970.

[Min61] M. Minsky. Recursive unsolvability of Post's problem of "Tag" and other topics in theory of Turing machines. *Annals of Math.*, 74(3), 1961.

[Pre29] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du I congrs de Mathmaticiens des Pays Slaves. Warsaw*, pages 92–101, 1929.

[Reu90] C. Reutenauer. *The Mathematics of Petri Nets*. Prentice-Hall, Inc., 1990.

[Rob49] J. Robinson. Definability and decision problems in arithmetic. *Journal of Symbolic Logic*, 14(2):98–114, 1949.

[Stu00] T. Sturm. Linear problems in valued fields. *J. Symb. Comput.*, 30(2):207–219, 2000.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[Tor91] J. Torán. Complexity classes defined by counting quantifiers. *J. ACM*, 38(3):753–774, 1991.

[Var01] M. Vardi. Branching vs. linear time: Final showdown. In *Proceedings of TACAS (LNCS Volume 2031)*, pages 1–22. Springer-Verlag, 2001.

[vzGS78] J. von zur Gathen and M. Sieveking. A bound on solutions of linear integer equalities and inequalities. *Proceedings of the American Mathematical Society*, 72(1):155–158, 1978.

[Wag86] K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Inf.*, 23(3):325–356, 1986.

[Wei88] V. Weispfenning. The complexity of linear problems in fields. *J. Symb. Comput.*, 5(1-2):3–27, 1988.

[Wol01] P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In *Summer School on Trends in Computer Science, 2000*, volume 2090 of *LNCS*, pages 261–277. Springer, 2001.