HAL open science

Rudiments of Presburger Arithmetic

Stéphane Demri

▶ To cite this version:

Stéphane Demri. Rudiments of Presburger Arithmetic. Master. Paris, France. 2016, pp.44. hal-03188114

HAL Id: hal-03188114
https://hal.science/hal-03188114v1

Submitted on 1 Apr 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1

RUDIMENTS OF PRESBURGER ARITHMETIC

1.1	**A Logical Formalism For Arithmetical Constraints**	**1**
1.2	**From Formulae to Interpretations**	**3**
1.3	**Quantifier Elimination**	**16**
1.4	**Automata-based Approach for Presburger Arithmetic**	**21**
1.5	**Semilinear Sets**	**25**
1.6	**Application: Parikh Image of Regular Languages**	**29**

(Lectures given on 09/10/15, 16/10/15, 30/09/16 and 07/10/16 by S. Demri)

1.1 A Logical Formalism For Arithmetical Constraints

Operations in counter machines induce constraints between the current values of counters and their next values, for instance x++ stands for an increment and it can be represented by the expression $\mathsf{x}' = \mathsf{x} + 1$ where x' is understood as the next value for the counter x. Similarly, "$\mathsf{x}' = \mathsf{x} + 1$ and $\mathsf{x} > \mathsf{y}$" can be understood as the same operation on the counter x except that we require that the guard $\mathsf{x} > \mathsf{y}$ is satisfied, namely x is currently strictly greater than y. Updates and guards used to define operations are expressed in a formal language that is able to state properties between natural numbers (and more generally between integers). *Presburger arithmetic* (Presburger, 1929), introduced by Presburger eighty years ago, can serve as a logical formalism for expressing constraints on integers. Roughly speaking, Presburger arithmetic is the first-order theory of the well-known structure $\langle \mathbb{N}, +, \leq \rangle$. Nowadays, it is ubiquitous in formal verification thanks to its numerous properties: decidability of the satisfiability problem (which contrasts with Peano arithmetic that includes multiplication) (Presburger, 1929), well-understood expressive power apart from being very expressive (defining exactly semilinear sets), Presburger arithmetic admits quantifier elimination and its quantifier-free fragment has a satisfiability problem with relatively low worst-case complexity, namely it is NP-complete. Finally, Presburger arithmetic can be understood as a standard first-order theory interpreted over the set of natural numbers. This makes it a handy formalism for anyone a bit familiar with

Presburger arithmetic

logical formalisms, but getting used to it does not require any serious prerequisites on logic.

There are numerous motivations for introducing Presburger arithmetic, the first-order theory of natural numbers with addition. Below, we pick some of them.

- Presburger arithmetic is a language that is useful to write programs. For instance, expressions from such a language can be used to write guards (such as "$x > 4$"), updates (such as "$x' = 2x$") or invariants (such as "$x+y = z$"), etc.. As in every programming language, a simple and clear syntax is required and it is essential that it is effective and executable. Presburger arithmetic takes advantage of the standard connectives in first-order logic (negation, conjunction, disjunction) as well as the universal and existential quantifiers. By way of example, the fact that x is even can be naturally expressed as "$\exists\, y\ (x = y+y)$" where $\exists$ is the standard existential quantifier.

- Presburger arithmetic is a mathematical formalism that is used in many technical developments in the course. Understanding its underlying theory (for instance the relationships between Presburger-definability and semilinearity) provides general and powerful tools and concepts (decidability, structural results, etc.). Each Presburger formula defines a set of tuples (the set of valuations that makes it true) and Presburger arithmetic is therefore a means to represent and manipulate symbolically infinite sets of tuples of natural numbers.

- Last but not least, decidability for Presburger arithmetic is used in numerous algorithms, not only for VASS by the way. These algorithms and their effective implementations are important concepts that are essential to be aware of. Algorithmic details often need to provide a refined analysis (size of constants, data structures for quantifier-free formulae, test in NP, etc.) that can go far beyond the simple decidability result.

In this chapter, we aim at presenting the main definitions and results as well as hints and proof ideas about the main results. Relevant and typical examples often replace a proof in due form.

As it will be apparent in the following, the objectives of this chapter are threefold. We introduce the first-order theory of natural numbers with addition, known as Presburger arithmetic; we define the syntax, semantics and we provide explanations about how to use such a formalism. We state its main properties and provide examples or intuitions for proofs (quantifier elimination, decidability, equivalence between semilinear sets and relations definable by Presburger arithmetic, complexity of Presburger arithmetic as well as for fragments). Finally, we present several applications that illustrate nicely how to take advantage of this well-known logical formalism.

1.2 From Formulae to Interpretations

We write $\mathbb{N}$ [resp. $\mathbb{Z}$] for the set of natural numbers [resp. integers] and $[m, m']$ with $m, m' \in \mathbb{Z}$ to denote the set $\{j \in \mathbb{Z} : m \leq j \leq m'\}$. For $\mathbf{x} \in \mathbb{Z}^n$, we write $\mathbf{x}(1), \ldots, \mathbf{x}(n)$ for the entries of $\mathbf{x}$. For $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, $\mathbf{x} \leq \mathbf{y} \overset{\text{def}}{\Leftrightarrow}$ for all $i \in [1, n]$, we have $\mathbf{x}(i) \leq \mathbf{y}(i)$. We also write $\mathbf{x} < \mathbf{y}$ when $\mathbf{x} \leq \mathbf{y}$ and $\mathbf{x} \neq \mathbf{y}$.

1.2.1 Syntax and semantics

In order to define the formal language for Presburger arithmetic, abbreviated by FO($\mathbb{N}$) in the sequel, we need to introduce several types of syntactic objects. The *formulae*, i.e. the expressions stating properties about the structure $\langle \mathbb{N}, +, \leq \rangle$ such as $\forall\ \mathsf{x}\ (\exists\ \mathsf{y}\ ((2\mathsf{x} + 8) \leq \mathsf{y})$, are built upon a set of *atomic formulae* and the full set of formulae is generated from atomic formulae by composing Boolean connectives and quantifiers. For instance, $(2\mathsf{x} + 8) \leq \mathsf{y}$ is an atomic formula in FO($\mathbb{N}$). Note that it is also made of *terms* such as y or $2\mathsf{x} + 8$. The terms itself are built from *variables* by using the addition operator '$+$'. A formula of the form

$$\forall\ \mathsf{x}\ (\exists\ \mathsf{y}\ ((2\mathsf{x} + 8) \leq \mathsf{y})$$

belongs to the logic FO($\mathbb{N}$) not only because of the way it is syntactically built but also because the variables are interpreted by natural numbers, the operator '$+$' is interpreted by the addition in $\langle \mathbb{N}, +, \leq \rangle$ and the symbol '$\leq$' is interpreted by the standard linear ordering on $\mathbb{N}$. All of this should sound very familiar to anyone acquainted with classical predicate logic and the definitions below can be viewed as an instance of such a standard logical formalism when interpreted over the structure $\langle \mathbb{N}, +, \leq \rangle$.

Now, let us define the formulae and then explain how to interpret such syntactic objects. Let VAR $= \{\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots\}$ be a countably infinite set of *variables*. The variables are interpreted below as natural numbers. *Terms* are built from constants in $\mathbb{N}$ and variables in VAR, and by making all the possible finite sums from constants and variables. Hence, terms are expressions of the form $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + k$ where $a_1, \ldots, a_n$ are constant coefficients in $\mathbb{N}$, k is in $\mathbb{N}$ and the x_i's are variables. When we write $a\mathsf{x}$ or k, we do not bother about the way the natural numbers a or k are encoded (in unary or in binary). This will have some importance only when computational complexity issues are considered and in that case, we shall be more explicit about the type of encoding. By default, we assume a binary encoding of integers.

Variables and terms come with their interpretations when the variables are interpreted by natural numbers. A *valuation* $\mathfrak{v}$ is a map VAR $\to \mathbb{N}$ and it can be extended to the set of all terms as follows:

- $\mathfrak{v}(k) = k$,
- $\mathfrak{v}(a\mathsf{x}) = a \times \mathfrak{v}(\mathsf{x})$ and

- $\mathfrak{v}(t + t') = \mathfrak{v}(t) + \mathfrak{v}(t')$ for all terms t and t'.

For instance, under the valuation $\mathfrak{v}$ such that $\mathfrak{v}(\mathsf{x}) = 3$ and $\mathfrak{v}(\mathsf{y}) = 27$, we can state that y is greater than $2\mathsf{x} + 8$, which can be written $\mathfrak{v} \models (2\mathsf{x} + 8) \leq \mathsf{y}$ where '$\models$' stands for the satisfaction relation. Indeed, $\mathfrak{v}(2\mathsf{x}+8) = 14$. By contrast, under the valuation $\mathfrak{v}$ such that $\mathfrak{v}(\mathsf{x}) = 3$ and $\mathfrak{v}(\mathsf{y}) = 13$, y is not greater than $2\mathsf{x} + 8$, which we write $\mathfrak{v} \not\models (2\mathsf{x} + 8) \leq \mathsf{y}$. Hence, a valuation is simply an interpretation for variables and terms that can make true or false relationships between terms.

Atomic formulae are built from terms by expressing relationships between two terms and they form the most elementary instances of formulae. For example, $(2\mathsf{x} + 8) \leq \mathsf{y}$ is an atomic formula and it cannot be broken down into strictly smaller (atomic) formulae. *Atomic formulae* are defined as expressions of the form $t \leq t'$. We also include the truth constants $\top$ (true) and $\bot$ (false) because this can be useful sometimes. The truth value of an atomic formula under a valuation $\mathfrak{v}$ is quite easy to define when '$\leq$' is interpreted as the standard ordering: $\mathfrak{v} \models t \leq t' \overset{\text{def}}{\Leftrightarrow} \mathfrak{v}(t) \leq \mathfrak{v}(t')$. Note that the first occurrence of '$\leq$' refers to a symbol used for defining atomic formulae whereas its second occurrence refers to the ordering relation on the set of natural numbers. Hence, that symbol is clearly overloaded but this should not cause any confusion in the following and we behave similarly with other well-understood symbols.

Formulae in $\mathrm{FO}(\mathbb{N})$ contains atomic formulae and more complex formulae can be generated by composition with negation (written $\neg$ as a unary connective), conjunction (written $\wedge$ as a binary connective) and disjunction (written $\vee$ as a binary connective). For example, here is a formula that states that $2\mathsf{x} + 8$ and y have identical value.

$$((2\mathsf{x} + 8) \leq \mathsf{y}) \wedge (\mathsf{y} \leq (2\mathsf{x} + 8)).$$

Boolean connectives such as negation, conjunction or disjunction allow to express richer properties than those by the atomic formulae only. In the following, we use abbreviations in order to simplify the presentation of formulae (t and t' are arbitrary terms):

$$\begin{array}{lcl} t = t' & \overset{\text{def}}{=} & (t \leq t') \wedge (t' \leq t) \\ t < t' & \overset{\text{def}}{=} & t + 1 \leq t' \\ t \geq t' & \overset{\text{def}}{=} & t' \leq t \\ t > t' & \overset{\text{def}}{=} & t' + 1 \leq t \end{array}$$

Formulae can be also obtained by using quantifiers such as '$\exists$' (existential quantifier) and '$\forall$' (universal quantifier). We have seen that $(2\mathsf{x} + 8) \leq \mathsf{y}$ is an atomic formula stating a property betwen x and y. We can build a formula by introducing an existential quantification providing the formula $\exists\, \mathsf{y}\ (2\mathsf{x} + 8) \leq \mathsf{y}$ that states a property about the variable x, namely that there exists a value for y for which $(2\mathsf{x} + 8) \leq \mathsf{y}$ holds true. Observe that the quantifier '$\exists$' is followed by a variable

(here y) over which the quantification is performed. It would not make sense to use a quantifier that is not followed by a variable. This process of building formulae can be repeated, for instance by adding a universal quantification, leading to a formula such as $\forall\, \mathsf{x}\, (\exists\, \mathsf{y}\, ((2\mathsf{x} + 8) \leq \mathsf{y}))$. This latter expression states a property about the structure $\langle \mathbb{N}, +, \leq \rangle$ rather than a property about variables: for every value for x, there is a value for y that satisfies $(2\mathsf{x} + 8) \leq \mathsf{y}$, which is a pretty valid statement on the structure $\langle \mathbb{N}, +, \leq \rangle$. So formulae in FO($\mathbb{N}$) are syntactic objects built from atomic formulae obtained by combining Boolean connectives such as $\neg$, $\wedge$ and $\vee$ and quantifiers such as $\exists$ and $\forall$. The ability to combine atomic formulae with logical connectives to generate more complex formulae explains why Presburger arithmetic is a very rich language, more about it can be found in the following. To sum up, *formulae* (usually written with the symbols φ, ψ or χ possibly decorated with exponents or subscripts) are defined by the grammar below:

$$\varphi ::= \top \mid \bot \mid t \leq t' \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists \mathsf{x}\, \varphi \mid \forall \mathsf{x}\, \varphi$$

where t and t' are terms and $\mathsf{x} \in \mathrm{VAR}$.

By way of example, here is a formula stating that there is an infinite number of multiples of three:

$$\forall\, \mathsf{x}\, (\exists\, \mathsf{y}\, (\mathsf{y} > \mathsf{x}) \wedge (\exists\, \mathsf{z}\, (\mathsf{y} = 3\mathsf{z}))).$$

The semantics for formulae in FO($\mathbb{N}$) is mainly defined with the help of the satisfaction relation (below written $\models$) that determines the conditions for the satisfaction of a formula under a given valuation. We write $\mathfrak{v} \models \varphi$ to denote that the formula φ holds true under the valuation $\mathfrak{v}$. We have already seen how to compute the truth value for $\mathfrak{v} \models t \leq t'$, i.e. when atomic formulae are involved. Below, we extend the definition of $\models$ to all formulae by taking advantage of the decomposition in terms of subformulae:

- $\mathfrak{v} \models \top \overset{\text{def}}{\Leftrightarrow}$ true,
- $\mathfrak{v} \models \bot \overset{\text{def}}{\Leftrightarrow}$ false,
- $\mathfrak{v} \models t \leq t' \overset{\text{def}}{\Leftrightarrow} \mathfrak{v}(t) \leq \mathfrak{v}(t')$,
- $\mathfrak{v} \models \neg\varphi \overset{\text{def}}{\Leftrightarrow}$ not $\mathfrak{v} \models \varphi$,
- $\mathfrak{v} \models \varphi \wedge \varphi' \overset{\text{def}}{\Leftrightarrow} \mathfrak{v} \models \varphi$ and $\mathfrak{v} \models \varphi'$,
- $\mathfrak{v} \models \varphi \vee \varphi' \overset{\text{def}}{\Leftrightarrow} \mathfrak{v} \models \varphi$ or $\mathfrak{v} \models \varphi'$,
- $\mathfrak{v} \models \exists\, \mathsf{x}\, \varphi \overset{\text{def}}{\Leftrightarrow}$ there is $n \in \mathbb{N}$ such that $\mathfrak{v}[\mathsf{x} \mapsto n] \models \varphi$ where $\mathfrak{v}[\mathsf{x} \mapsto n]$ is equal to $\mathfrak{v}$ except that x is mapped to n,
- $\mathfrak{v} \models \forall\, \mathsf{x}\, \varphi \overset{\text{def}}{\Leftrightarrow}$ for every $n \in \mathbb{N}$, we have $\mathfrak{v}[\mathsf{x} \mapsto n] \models \varphi$.

It is the place where the symbols $\neg$, $\wedge$, $\vee$, $\exists$ and $\forall$ are semantically bound and it is easy to verify that, indeed, $\neg$ is understood as a negation operator, $\exists$ is understood as an existential quantification etc. It is also the interpretation of the logical connectives that guarantees that $\mathfrak{v} \models t = t'$ (where '$t = t'$' is an abbreviation) iff $\mathfrak{v}(t) = \mathfrak{v}(t')$, making sense to the choice of the abbreviation. A similar reasoning can be done with the other abbreviations introduced earlier. More shortcuts will be introduced and used in this document but before doing so, let us present a simple example.

Example 1.1. Let φ_{tot} be the formula stating that $\langle \mathbb{N}, < \rangle$ is a linearly ordered set (abbreviations are used):

$$\varphi_{\text{tot}} \stackrel{\text{def}}{=} \forall\, \mathsf{x}\, \forall\, \mathsf{y}\, ((\mathsf{x} = \mathsf{y}) \vee (\mathsf{x} < \mathsf{y}) \vee (\mathsf{x} > \mathsf{y})).$$

It is worth noting that for every valuation $\mathfrak{v}$, we have

$$\mathfrak{v} \models (\mathsf{x} = \mathsf{y}) \vee (\mathsf{x} < \mathsf{y}) \vee (\mathsf{x} > \mathsf{y}),$$

which is the key argument to see that for every valuation $\mathfrak{v}$, we have $\mathfrak{v} \models \varphi_{\text{tot}}$.

The satisfaction relation is defined in such a way that formulae that are syntactically different can be satisfied by exactly the same valuations. For instance, $\exists\, \mathsf{x}\, \varphi$ and $\neg\forall\, \mathsf{x}\, \neg\varphi$ can be shown to be satisfied by exactly the same valuations, whatever the formula φ is. This witnesses the standard duality between existential quantification and universal quantification. Similarly, $\varphi_1 \wedge \varphi_2$ and $\neg(\neg\varphi_1 \vee \neg\varphi_2)$ are satisfied by exactly the same valuations (known as one of De Morgan's laws). In a sense, this illustrates a slight redundancy in our choice for logical connectives but is has the great advantage to use more concise formulae with, hopefully, a more straightforward understanding. These observations motivate the following definition: φ and ψ are *equivalent* in $\text{FO}(\mathbb{N}) \stackrel{\text{def}}{\Leftrightarrow}$ for every valuation $\mathfrak{v}$, we have $\mathfrak{v} \models \varphi$ iff $\mathfrak{v} \models \psi$. So, $\exists\, \mathsf{x}\, \varphi$ and $\neg\forall\, \mathsf{x}\, \neg\varphi$ are equivalent formulae. Similarly, $\forall\, \mathsf{x}\, \exists\, \mathsf{y}\, (\mathsf{y} < \mathsf{x})$ and $\forall\, \mathsf{x}\, \exists\, \mathsf{y}\, (\mathsf{x} \leq \mathsf{y})$ can be shown as non-equivalent.

We have defined formally how to construct terms, atomic formulae and formulae and there is a clear discipline to generate such syntactic objects. However, in some places, we might be inclined to write expressions a bit differently in order to comply with standard mathematical and logical practice or to use shortcuts avoiding lengthy developments. Below, we list standard abbreviations.

- A formula of the form $\forall\, \mathsf{x}_1 \cdots \forall\, \mathsf{x}_n\, \varphi$ is also written

$$\forall\, \mathsf{x}_1, \ldots, \mathsf{x}_n\, \varphi.$$

- We make use of standard Boolean connectives that can be defined from negation, conjunction and disjunction such as material implication and equivalence.

- The expression $\varphi \Rightarrow \psi$ is an abbreviation for the formula $(\neg\varphi) \vee \psi$. It is easy to check that for every valuation $\mathfrak{v}$, $\mathfrak{v} \models \varphi \Rightarrow \psi$ iff ($\mathfrak{v} \models \varphi$ implies $\mathfrak{v} \models \psi$).
- Analogously, the expression $\varphi \Leftrightarrow \psi$ is an abbreviation for the formula $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$. Note that for every valuation $\mathfrak{v}$, $\mathfrak{v} \models \varphi \Leftrightarrow \psi$ iff ($\mathfrak{v} \models \varphi$ iff $\mathfrak{v} \models \psi$). So, φ and ψ are equivalent formulae whenever for every valuation $\mathfrak{v}$, we have $\mathfrak{v} \models \varphi \Leftrightarrow \psi$.

- Another useful abbreviation consists in writing

$$\forall_{\leq k}\, \mathsf{x}\, \varphi$$

with some $k \geq 0$ in place of $\forall\, \mathsf{x}\, (\mathsf{x} \leq k) \Rightarrow \varphi$.

The intended semantics of such a construction is indeed clear since for every valuation $\mathfrak{v}$, $\mathfrak{v} \models \forall_{\leq k}\, \mathsf{x}\, \varphi$ iff for every $n \leq k$, we have $\mathfrak{v}[\mathsf{x} \mapsto n] \models \varphi$. Similarly, we write $\exists_{\leq k}\, \mathsf{x}\, \varphi$ with some $k \geq 0$ in place of $\exists\, \mathsf{x}\, (\mathsf{x} \leq k) \wedge \varphi$.

- Strictly speaking, the construction of the terms does not allow to use negative integers such as in the expression $-2\mathsf{x}+3\mathsf{y}-8$. Indeed, the summation of variables and the summation of the constant one do not permit to obtain negative coefficients. However, in atomic formulae, we shall permit the use of negative coefficients because it is indeed equivalent to an expression respecting strictly our syntax. For instance, the expression $-2\mathsf{x}+3\mathsf{y}-8 \leq 5\mathsf{x}$ is understood as the atomic formula $3\mathsf{y} \leq 7\mathsf{x}+8$. Similarly, terms are understood modulo associativity and commutativity: for instance, in a formula we do not distinguish the term $7\mathsf{x}+8$, from $3\mathsf{x}+8+4\mathsf{x}$ or $2+6\mathsf{x}+6+\mathsf{x}$. In the following, the very presentation of specific atomic formulae and terms is guided by the most handy way to express them.
- Let us conclude this part about abbreviations by introducing a shortcut to express *modulo constraints*. We have seen that the formula $\exists\, \mathsf{y}\, (\mathsf{x} = 3\mathsf{y})$ states that x is interpreted by a multiple of three. It is clearly more convenient if we could simply write $\mathsf{x} \equiv_3 0$. That is the reason why, we introduce the following shortcut. We write $t \equiv_k t'$ where t and t' are arbitrary terms and $k \geq 0$, to denote the formula

$$\exists\, \mathsf{x}\, (t = k\mathsf{x} + t') \vee (t' = k\mathsf{x} + t)$$

where x is a new variable not occurring in t and t'. Note that the cases when k takes the value 0, $\equiv_0$ corresponds to the equality and $t \equiv_1 t'$ corresponds to $\top$. Modulo constraints are not only handy macros but they will play also an important role in Section 1.3 when dealing with quantifier elimination.

The combination of all these abbreviations allows us to write the expression below as a formula from FO($\mathbb{N}$):

$$\forall\, \mathsf{x}, \mathsf{y}\, (-2\mathsf{x} + 9 \equiv_4 \mathsf{y} + 1) \Leftrightarrow (-\mathsf{y} \equiv_4 2\mathsf{x} - 8)$$

1.2.2 Decision problems

The main decision problem for FO($\mathbb{N}$) is to determine whether a formula admits a valuation that satisfies it. The atomic formula $\mathsf{x} + 3 \leq \mathsf{y}$ admits such a valuation $\mathfrak{v}$, for instance by requiring that $\mathfrak{v}(\mathsf{x}) = 12$ and $\mathfrak{v}(\mathsf{y}) = 2^{100}$. The interpretation of the variables different from x and y is not really relevant to check whether a valuation satisfies $\mathsf{x} + 3 \leq \mathsf{y}$. Note also that the atomic formula can certainly be satisfied by a valuation for which the values for $\mathfrak{v}(\mathsf{x})$ and $\mathfrak{v}(\mathsf{y})$ are much smaller. So the *satisfiability problem* for Presburger arithmetic is a decision problem that takes as input a formula φ and asks whether there is a valuation $\mathfrak{v}$ such that $\mathfrak{v} \models \varphi$. If such a valuation exists, we say that φ is *satisfiable*. By way of example, let us consider the formula φ_{exp} below:

$$\varphi_{\text{exp}} \stackrel{\text{def}}{=} (\mathsf{x}_1 \geq 2) \wedge (\mathsf{x}_2 \geq 2\mathsf{x}_1) \wedge \cdots \wedge (\mathsf{x}_n \geq 2\mathsf{x}_{n-1})$$

It is clearly satisfiable but any valuation that satisfies it requires at least that the interpretation of x_n is greater than 2^n even though the formula is only of size $\mathcal{O}(n)$.

Satisfiability comes with its dual concept known as *validity* so that the existential quantification over valuations in the definition for the satisfiability problem is replaced by a universal quantification. The *validity problem* for Presburger arithmetic is a decision problem that takes as input a formula φ and asks whether for every valuation $\mathfrak{v}$, we have $\mathfrak{v} \models \varphi$. If φ is satisfied by every valuation, we say that φ is *valid*.

The above formula φ_{exp} is clearly not valid since any valuation such that x_1 is interpreted by zero falsifies φ_{exp}. By contrast, the formula below is valid:

$$(\mathsf{x}_1 \geq 2 \wedge \mathsf{x}_2 \geq 2\mathsf{x}_1 \wedge \cdots \wedge \mathsf{x}_n \geq 2\mathsf{x}_{n-1}) \Rightarrow \mathsf{x}_n \geq 2^n.$$

Even though validity and satisfiability problems are first-class problems for FO($\mathbb{N}$), below we explain the close relationships between them so that in the sequel we do not need to really consider validity at all. For instance, it is straightforward that every valid formula is satisfiable since a valid formula is satisfied by *every* valuation. We have also seen that satisfiability does not imply validity (see e.g. the formula φ_{exp}), which should not come as a surprise since satisfiability asks for the satisfaction by a *single* valuation. There are also formulae for which satisfiability is equivalent to validity and it is the purpose of the following paragraph.

An occurrence of the variable x in the formula φ is *free* if it does not occur in the scope of either $\exists\, \mathsf{x}$ or $\forall\, \mathsf{x}$. For instance, all the occurrences of the variables in φ_{exp} are free since it does not contain any quantification. Otherwise, the occurrence is *bound*. For instance, in $\mathsf{x}_1 < \mathsf{x}_2$, all the occurrences of the variables are free. In $(\exists\, \mathsf{x}_1, \mathsf{x}_2\ (\mathsf{x}_1 < \mathsf{x}_2)) \wedge \mathsf{x}_1 < \mathsf{x}_2$, each variable among x_1, x_2, has a free occurrence and a bound occurrence. Given a formula φ, we write $free(\varphi)$ to denote the free variables occurring in φ.

Lemma 1.2 below states how to reduce simply an instance of the validity problem to an instance of the satisfiability problem and the other way around. Quantifications in formulae (in the object language) are used to perform quantifications over valuations (in the meta-language).

Lemma 1.2. *Let φ be a formula whose free variables are among $\mathsf{x}_1, \ldots, \mathsf{x}_n$. The propositions below are equivalent:*

(I) *φ is valid.*

(II) *$\forall\, \mathsf{x}_1, \ldots, \mathsf{x}_n\ \varphi$ is valid;*

(III) *$\forall\, \mathsf{x}_1, \ldots, \mathsf{x}_n\ \varphi$ is satisfiable.*

(IV) *$\forall\, \mathsf{x}_1, \ldots, \mathsf{x}_n\ \varphi$ is equivalent to $\top$.*

Similarly, the propositions below are equivalent:

(V) *φ is satisfiable.*

(VI) *$\exists\, \mathsf{x}_1, \ldots, \mathsf{x}_n\ \varphi$ is valid.*

(VII) *$\exists\, \mathsf{x}_1, \ldots, \mathsf{x}_n\ \varphi$ is satisfiable.*

(VIII) *$\exists\, \mathsf{x}_1, \ldots, \mathsf{x}_n\ \varphi$ is equivalent to $\top$.*

The proof of Lemma 1.2 is left as an exercise and it uses the duality between the quantifiers $\forall$ and $\exists$. That is why, in the sequel, we are mainly interested in the satisfiability problem. Moreover, when a formula has no free variable occurrences, satisfiability is equivalent to validity and therefore the difference is even less relevant.

We write $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ to denote a formula with free variables among $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$, for some $n \geq 1$. The formula defines the following set of n-tuples:

$$[\![\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)]\!] \stackrel{\text{def}}{=} \{\langle \mathfrak{v}(\mathsf{x}_1), \ldots, \mathfrak{v}(\mathsf{x}_n)\rangle \in \mathbb{N}^n : \mathfrak{v} \models \varphi\}.$$

The set $[\![\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)]\!]$ contains all the tuples that make true the formula φ by ignoring the irrelevant interpretation of the bound variables and by fixing an arbitrary ordering between the variables. In that respect, using the notation $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ has the advantage to provide an ordering of the variables. When $\mathsf{x}_1, \ldots, \mathsf{x}_n$ is known without ambiguity, the set $[\![\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)]\!]$ is simply denoted by $[\![\varphi]\!]$. For instance, $[\![\mathsf{x}_1 < \mathsf{x}_2]\!] = \{\langle n, n'\rangle \in \mathbb{N}^2 : n < n'\}$. Similarly, the set of odd natural numbers can be defined by the formula below:

$$\exists \mathsf{y}\ \mathsf{x} = \mathsf{y} + \mathsf{y} + 1.$$

The set $\{0\}$ can be defined by the formula $\mathsf{x} = \mathsf{x} + \mathsf{x}$. Similarly, the set $\{1\}$ is defined by $\exists\, \mathsf{y}\ (\mathsf{y} = \mathsf{y} + \mathsf{y} \wedge \mathsf{x} = \mathsf{y} + 1)$.

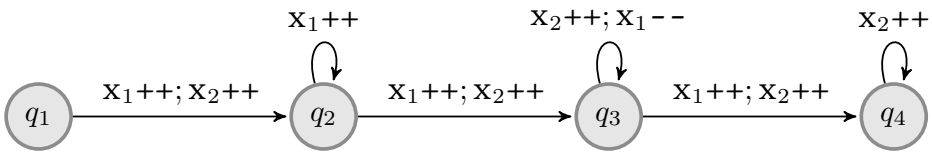

Figure 1.1: A simple counter machine

Moreover satisfiability of φ is equivalent to the nonemptiness of $[\![\varphi]\!]$. Similarly, φ is valid iff $[\![\varphi]\!] = \mathbb{N}^n$.

Let us conclude this section by a definition for sets in $\mathbb{N}^n$ that can be defined from formulae in FO($\mathbb{N}$).

Definition 1.3 (Presburger set)**.** Let φ be a formula $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ with $n \geq 1$ free variables among $\mathsf{x}_1, \ldots, \mathsf{x}_n$. We say that $[\![\varphi]\!]$ is a *Presburger set*.

The characterization of Presburger sets by means different from the use of FO($\mathbb{N}$) is discussed in Section 1.5. It is clear that Presburger sets are closed under intersection, union and complementation because of the presence of Boolean connectives in the logical formalism FO($\mathbb{N}$).

By way of example, let us consider the counter machine presented in Figure 1.1 with two counters and four control states.

We write X_i to denote the set $\{\langle n, m\rangle \mid \langle q_1, 0, 0\rangle \xrightarrow{*} \langle q_i, n, m\rangle\}$ for every $i \in [1, 4]$. One can show that all the sets $X_1, \ldots, X_4$ included in $\mathbb{N}^2$ are Presburger sets; more precisely

$$\begin{array}{lcl} X_1 & = & [\![\mathsf{x}_1 = \mathsf{x}_2 = 0]\!] \\ X_2 & = & [\![\mathsf{x}_2 = 1 \wedge \mathsf{x}_1 \geq 1]\!] \\ X_3 & = & [\![\mathsf{x}_2 \geq 2 \wedge \mathsf{x}_1 + \mathsf{x}_2 \geq 4]\!] \\ X_4 & = & [\![\mathsf{x}_1 \geq 1 \wedge \mathsf{x}_2 \geq 3 \wedge \mathsf{x}_1 + \mathsf{x}_2 \geq 6]\!]. \end{array}$$

An alternative formula (with quantifiers) to define X_4 is provided below. For each self-loop, a quantifier variable is introduced that corresponds to the number of times the self-loop is visited.

$$\exists\, \mathsf{z}_1, \mathsf{z}_2, \mathsf{z}_3\ (\mathsf{x}_1 = 3 + \mathsf{z}_1 - \mathsf{z}_2) \wedge (\mathsf{x}_2 = 3 + \mathsf{z}_2 + \mathsf{z}_3) \wedge\ (2 + \mathsf{z}_1 - \mathsf{z}_2 \geq 0).$$

1.2.3 Fragments

In this section, we introduce several syntactic fragments of FO($\mathbb{N}$) by restricting the use of syntactic resources or by imposing constraints on the form of the terms, formulae, etc.

First, we need to provide a few more explanations about the formal differences between abbreviations and built-in predicate symbols or atomic formulae. Above, the syntax for terms, atomic formulae and formulae in FO($\mathbb{N}$) has been extended in order to admit several abbreviations that are helpful to write formulae. For

instance, $(\mathsf{x} = \mathsf{y}) \Rightarrow (\mathsf{y} = \mathsf{x})$ use abbreviations and strictly speaking, should be understood as the formula below:

$$\neg((\mathsf{x} \leq \mathsf{y}) \wedge (\mathsf{y} \leq \mathsf{x})) \vee ((\mathsf{y} \leq \mathsf{x}) \wedge (\mathsf{x} \leq \mathsf{y})).$$

Indeed, '$\Rightarrow$' is not a built-in logical connective whereas '$=$' is not the built-in equality predicate symbol; these symbols are used for the sake of conciseness and simplicity. It is also possible to consider these symbols as primitive in the syntax by extending the set of logical connectives or by extending the set of atomic formulae. This would not make any essential difference with the current version since material implication or equalities can be expressed in it.

By contrast, the use of new built-in symbols would make a difference when fragments of FO($\mathbb{N}$) are involved because we do not have access to the full power of the logical formalism. By way of example, let us consider the expression $t \equiv_k t'$ that is a shortcut for

$$\exists\, \mathsf{x}\, (t = k\mathsf{x} + t') \vee (t' = k\mathsf{x} + t).$$

Even though $t \equiv_k t'$ does not make apparent the presence of quantifiers, it does contain an existential quantification. However, it is also possible to extend the set of formulae by considering $t \equiv_k t'$ as a new primitive atomic formula with the following semantics: $\mathfrak{v} \models t \equiv_k t' \overset{\text{def}}{\Leftrightarrow} \mathfrak{v}(t)$ is equal to $\mathfrak{v}(t')$ modulo k. In that latter case, $t \equiv_k t'$ is quantifier-free. Below, we define fragments of FO($\mathbb{N}$) by imposing syntactic restrictions but at the same time, we add new primitive predicate symbols. Unless otherwise stated, we assume that the set of primitive atomic formulae is extended as follows:

$$\top \mid \bot \mid t \leq t' \mid t \equiv_k t' \mid t = t' \mid t < t' \mid t \geq t' \mid t > t' \qquad \text{(PAF)}$$

with $k \geq 2$. The semantics of the new binary predicate symbols is without any surprise, for instance $\mathfrak{v} \models t > t' \overset{\text{def}}{\Leftrightarrow} \mathfrak{v}(t) > \mathfrak{v}(t')$. We know that adding these symbols does not increase the expressive power of FO($\mathbb{N}$) since we were able to find equivalent formulae in the original version of FO($\mathbb{N}$).

From now on, we assume that FO($\mathbb{N}$) contains all the above-mentioned atomic formulae.

Definition 1.4 (Quantifier-free fragment)**.** A formula φ is quantifier-free $\overset{\text{def}}{\Leftrightarrow}$ φ is a Boolean combination of atomic formulae (i.e. without quantifiers).

For instance $(\mathsf{x} + \mathsf{y} \equiv_5 \mathsf{z}) \vee (\mathsf{y} > 23)$ is a quantifier-free formula that contains exactly four occurrrences of free variables. It is clear that, in a quantifier-free formula, all the variable occurrences are free.

Definition 1.5 (Linear fragment)**.** A formula φ is in the linear fragment $\overset{\text{def}}{\Leftrightarrow}$ φ is a Boolean combination of atomic formulae of one of the forms below:

$$\top \mid \bot \mid t \leq t' \mid t = t' \mid t < t' \mid t \geq t' \mid t > t' \qquad \text{(LIN)}$$

So, φ is in the linear fragment of FO($\mathbb{N}$) when it is in the quantifier-free fragment and it does not contain any periodicity constraint.

Definition 1.6 (Difference fragment)**.** A formula φ is in the difference fragment $\overset{\text{def}}{\Leftrightarrow}$ φ belongs to the linear fragment and the terms are either of the form $\mathsf{x} + k$ or k.

For instance $\neg(\mathsf{x} = \mathsf{y}+8) \wedge \mathsf{y} \geq 7$ belongs to the difference fragment whereas $2\mathsf{x} = 6$ or $\mathsf{x} + \mathsf{y} \geq 3$ do not. Indeed, the difference fragment can only compare a variable or a *difference between two variables* with a constant value.

Let us conclude the presentation of fragments, by introducing a fragment of FO($\mathbb{N}$) in which quantifications are performed at the beginning of the formula.

Definition 1.7 (Prenex normal form)**.** A formula φ in FO($\mathbb{N}$) is in *prenex normal form* whenever it is of the form below

$$\mathcal{Q}_1\ \mathsf{x}_1\ \cdots \mathcal{Q}_n\ \mathsf{x}_n\ \psi$$

where ψ is in the linear fragment and $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_n\} \subseteq \{\exists, \forall\}$. $\mathcal{Q}_1\ \cdots \mathcal{Q}_n$ is called the *quantifier prefix* and its quantifier alternation is the number of positions i such that $\mathcal{Q}_i$ and $\mathcal{Q}_{i+1}$ are different.

Even though formulae in prenex normal form strongly restrict the use of quantifiers, one can show that every formula in FO($\mathbb{N}$) is equivalent to a formula in prenex normal form. The proof is left as an exercise and it is very similar to the proof of the similar property for classical predicate logic. A first step in the proof consists in showing that without any loss of generality, we can assume that two distinct quantifier occurrences such as $\mathcal{Q}\ \mathsf{x}$ and $\mathcal{Q}'\ \mathsf{x}'$ in the original formula verifies that x is a variable different from x'. Duality between quantifiers as well as duality between logical connectives need also to be used.

Let us consider the formula φ below:

$$\neg(\exists\ \mathsf{x}\ \mathsf{x} \geq 3) \vee (\forall\ \mathsf{y}\ \mathsf{y} \geq 4).$$

φ is equivalent to the following formula in prenex normal form:

$$\forall\ \mathsf{x}\ \forall\ \mathsf{y}\ (\neg(\mathsf{x} \geq 3) \vee \mathsf{y} \geq 4).$$

Definition 1.8 (Extended prenex normal form)**.** A formula φ in FO($\mathbb{N}$) is in *extended prenex normal form* whenever it is of the form below

$$(\mathcal{Q}_1)_{\leq k_1}\ \mathsf{x}_1\ \cdots (\mathcal{Q}_n)_{\leq k_n}\ \mathsf{x}_n\ \psi$$

where ψ is in the linear fragment, $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_n\} \subseteq \{\exists, \forall\}$ and $k_1, \ldots, k_n \in \mathbb{N}$.

1.2.4 Decidablility of the satisfiability problem

Designing a decision procedure for checking the satisfiability problem for FO($\mathbb{N}$) is known to be a difficult task because the domain of interpretation for variables is infinite (namely, the set of natural numbers) and therefore quantifications are performed over an infinite set. By comparison, designing a decision procedure for propositional calculus is much easier because each propositional variable is interpreted by either true or false (or equivalently by either 1 or 0) and therefore there is a finite set of relevant propositional valuations for checking the satisfiability status of a given propositional formula. Moreover, checking whether a propositional valuation satisfies a propositional formula can be done in polynomial time, which guarantees the existence of a simple algorithm for checking satisfiability (even though it may require exponential time in the worst case).

By contrast, checking whether a valuation $\mathfrak{v}$ satisfies a formula φ, even when the valuation is restricted to the variables occurring in φ is as hard as the satisfiability problem itself. Indeed, suppose that φ contains the variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$. We have φ is satisfiable iff $\mathfrak{v}_0 \models \exists\, \mathsf{x}_1, \ldots, \mathsf{x}_n \varphi$ where $\mathfrak{v}_0$ is a valuation such that all the variables from φ are interpreted by zero.

Quantifier-free fragment of FO($\mathbb{N}$) admits a simpler satisfiability problem since quantifications are disallowed but still one needs to determine whether a valuation can satisfy the formula. Assume that terms in quantifier-free formulae can be written as $(\sum_i a_i \mathsf{x}_i) + k$ where the a_i's and k belong to $\mathbb{N}$ and the natural numbers are encoded in binary. This is indeed a place where we have to be precise about the encodings but note that the current option is to encode concisely the natural numbers. The size of a formula, written $|\varphi|$, is roughly defined as the number of symbols occuring in it, which is a reasonably succinct encoding.

Theorem 1.9. *Let φ be a quantifier-free formula with variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$. The formula φ is satisfiable iff there is a valuation satisfying φ and each variable is interpreted by a natural number whose value is bounded by $2^{p(|\varphi|)}$ where $p(\cdot)$ is a polynomial independent of φ and $\mathsf{x}_1, \ldots, \mathsf{x}_n$.*

Theorem 1.9 states that to determine the satisfiability status of a quantifier-free formula, it is sufficient to inspect values interpreting variables below a bound $2^{p(|\varphi|)}$ (see bibliographical references in Section 1.6.2). It is possible to refine this bound by taking into account in a more precise way, the number of variables, the maximal size of a constant occurring in φ or the number of connective occurrences with a conjunctive polarity. Moreover, this is sufficient to prove the NP upper bound.

Corollary 1.10. *The satisfiability problem for the quantifier-free fragment of* FO($\mathbb{N}$) *is NP-complete.*

NP-hardness can be easily shown by defining a straightforward reduction from the satisfiability problem for propositional calculus known as SAT. Indeed,

given a formula φ with propositional variables $p_1, \ldots, p_n$, one can build a quantifier-free formula φ' obtained from φ by replacing p_i by $\mathsf{x}_i^{\text{new}} = \mathsf{y}_i^{\text{new}}$ such that φ is satisfiable iff φ' is satisfiable. The variables $\mathsf{x}_i^{\text{new}}$'s and $\mathsf{y}_i^{\text{new}}$'s are new and they serve the purpose of encoding propositional variables.

In order to obtain the NP upper bound, it is sufficient to guess

$$\langle \alpha_1, \ldots, \alpha_n \rangle \in [0, 2^{p(|\varphi|)}]^n$$

and then check that $\mathfrak{v} \models \varphi$ where $\mathfrak{v}(\mathsf{x}_i) = \alpha_i$ for every $i \in [1, n]$. Such a check should be done in polynomial time in the size of the formula, which is the case for the following reasons.

1. $\langle \alpha_1, \ldots, \alpha_n \rangle$ is of polynomial size in $|\varphi|$.
2. Computing $\mathfrak{v}(t)$ for any term t in φ can be done in polynomial time in $|\varphi|$.
3. Similarly, determining the truth value of any atomic formula under $\mathfrak{v}$ can be done in polynomial time in $|\varphi|$.
4. Finally, replacing all the atomic formulae from φ by either $\top$ or $\bot$ (depending on the truth value under $\mathfrak{v}$) and then simplifying using the semantics of Boolean connectives, leads to $\top$ or $\bot$ and this can be done in polynomial time.

Hence, the verification phasis for $\mathfrak{v} \models \varphi$ requires only polynomial time. Even though the values in $\mathfrak{v}$ can be of exponential magnitude with respect to the size of φ, their encoding requires only polynomial space which is useful to perform the final check in polynomial time.

Let us come back to our original main question. What about the decidability status of the satisfiability problem for FO($\mathbb{N}$)? It would be nice if Theorem 1.9 can be adapted to full Presburger arithmetic. Actually, it can but with much higher upper bounds as stated later on (see Theorem 1.12). Indeed, the key argument is to show that every formula φ admits an equivalent quantifier-free formula φ' such that φ' can be effectively built from φ. Equivalence implies that the satisfiability status of formulae is identical. Since the reduction is effective, we can then invoke Corollary 1.10 and get decidability.

Theorem 1.11 (Decidability). *The satisfiability problem for* FO($\mathbb{N}$) *is decidable.*

It should not come as a surprise that the elimination of quantifiers is computationally expensive and the construction of φ' from φ is done via a lengthy process. Section 1.3 is actually dedicated to quantifier elimination.

We have seen that disjunctions can be defined from conjunctions and negations, and therefore the connective for disjunction $\vee$ is a built-in connective that does not add any expressive power to FO($\mathbb{N}$). Similarly, quantifier elimination implies that the built-in quantifiers do not add any expressive power to FO($\mathbb{N}$)

when the atomic formulae are of the form either $t \leq t'$ or $t \equiv_k t'$. Indeed, periodicity constraints enter into the play here and this is necessary as briefly explained in Section 1.3. So, even though universal and existential quantifiers do not add any new expressive power to the quantifier-free fragment, such binders are useful to express concisely properties on natural numbers. This is also reflected by the fact that the satisfiability problem for FO($\mathbb{N}$) is clearly not in NP and somewhere between 2ExpTime and 2ExpSpace. More precisely, the satisfiability problem for FO($\mathbb{N}$) is complete for the class of alternating Turing machines working in double exponential time with at most a linear amount of alternations. This looks like a scary statement but this provides a complete complexity characterization and most of the time, we will be happy with the respective bounds 2ExpTime and 2ExpSpace.

One can also observe that decidability is a straightforward consequence of quantifier elimination and there is no need to invoke the NP-completeness of the quantifier-free fragment. Indeed, let φ be a formula whose free variables are among $\mathsf{x}_1, \ldots, \mathsf{x}_n$. We have seen that φ is satisfiable iff $\exists\, \mathsf{x}_1, \ldots, \mathsf{x}_n\ \varphi$ is equivalent to $\top$ (see Lemma 1.2). Satisfiability of φ is checked by eliminating quantifiers in the formula $\exists\, \mathsf{x}_1, \ldots, \mathsf{x}_n\ \varphi$ and verify that the quantifier elimination procedure leads to $\top$. Hence, quantifier elimination is a sufficiently strong property to entail decidability.

The procedure for eliminating quantifiers, and its improvements, is also at the heart of optimal complexity results for checking satisfiability, as stated below.

Theorem 1.12. *Let φ be a sentence in* FO($\mathbb{N}$) *(no free variables) of the form*

$$\mathcal{Q}_1\ \mathsf{x}_1\ \cdots\ \mathcal{Q}_s\ \mathsf{x}_s\ \psi(\mathsf{x}_1, \ldots, \mathsf{x}_s)$$

of length n and with m quantifier alternations such that ψ is quantifier-free. Then, in the quantification we can restrict ourselves to values bounded by $w = 2^{\mathsf{C} \times n^{[(s+3)^{m+2}]}}$ for some constant C*. This means that φ is satisfiable iff*

$$(\mathcal{Q}_1)_{\leq w}\ \mathsf{x}_1\ \cdots\ (\mathcal{Q}_s)_{\leq w}\ \mathsf{x}_s\ \psi(\mathsf{x}_1, \ldots, \mathsf{x}_s)$$

is satisfiable.

Bounding the quantifications allows to decide the satisfiability status of the formula by running over all the possible values for the variables and check how the matrix formula $\psi(\mathsf{x}_1, \ldots, \mathsf{x}_s)$ is satisfied. A bit of care is needed because quantifiers are either existential or universal but a simple nondeterministic algorithm can easily handle that.

Variants. Let FO($\mathbb{Z}$) be the variant of FO($\mathbb{N}$) in which variables are interpreted in $\mathbb{Z}$ and quantifications are performed over $\mathbb{Z}$. FO($\mathbb{Z}$) share with FO($\mathbb{N}$) all its nice properties: decidability, quantifier elimination, complexity, etc.

Theorem 1.13. *The satisfiability problem for* $\mathrm{FO}(\mathbb{Z})$ *is decidable.*

Exercise 1.15 presents a reduction from the satisfiability for $\mathrm{FO}(\mathbb{Z})$ to the satisfiability for $\mathrm{FO}(\mathbb{N})$ by proposing a simple encoding of integers within the natural numbers (parity of a natural number determines whether it encodes a negative integer).

Besides, it is also worth noting that the first-order theory of $\langle \mathbb{N}, \leq, \times \rangle$ is decidable too (known as *Skolem arithmetic*) whereas the first-order theory of $\langle \mathbb{N}, \leq, \times, + \rangle$ is undecidable.

1.3 Quantifier Elimination

In this section, we briefly explain what is quantifier elimination and how it can be proved for $\mathrm{FO}(\mathbb{N})$. Roughly speaking, quantifier elimination means that every formula in $\mathrm{FO}(\mathbb{N})$ admits an equivalent formula without quantifiers, i.e. quantifiers can be eliminated. A quantifier-free formula can be viewed as a Boolean combination of atomic formulae of the form $t \leq t'$, $t \equiv_k t'$, $\top$ or $\bot$. Moreover, we require that the equivalent quantifier-free variant has no more free variables than the original formula. In particular, this means that if the original formula is a sentence, i.e. without free variable occurrences, then its equivalent formula is either $\top$ or $\bot$.

1.3.1 The main statement

At first glance, it is rather disturbing to discover that the quantifier-free fragment of $\mathrm{FO}(\mathbb{N})$ is as expressive as full $\mathrm{FO}(\mathbb{N})$. Indeed, this might imply that quantifications in $\mathrm{FO}(\mathbb{N})$ are dummy and that they do not serve any purpose. This is not quite the case because eliminating quantifiers can be computationnally expensive and therefore the use of quantifiers is justified by the need to write concise formulae to express simple properties. This is different from the situation in which disjunction can be expressed by negation and conjunction. A better analogy would be to note that linear-time temporal logic LTL and first-order logic on ω-sequences are equally expressive logical formalisms by Kamp's Theorem but the two formalisms have distinct conciseness and admit satisfiability problems with very different complexity (PSpace-completeness versus non-elementarity in the present case).

As a starter, let us provide a few examples of formulae in which quantifiers can be (easily) eliminated:

- $\exists\, \mathsf{x}\ (\mathsf{x} \geq 3)$ is equivalent to $\top$,
- $\exists\, \mathsf{z}\ (\mathsf{x} < \mathsf{z} \wedge \mathsf{z} < \mathsf{y})$ is equivalent to $\mathsf{x} + 2 \leq \mathsf{y}$,
- $\exists\, \mathsf{z}\ (\mathsf{x} < \mathsf{z} \vee \mathsf{z} < \mathsf{y})$ is equivalent to $\top$,

- $\forall\, \mathsf{z}\ (\mathsf{x} \leq \mathsf{z} \Rightarrow \mathsf{y} \leq \mathsf{z})$ is equivalent to $\mathsf{y} \leq \mathsf{x}$,
- $\exists\, \mathsf{z}\ \mathsf{x} = 2\mathsf{z}$ is equivalent to $\mathsf{x} \equiv_2 0$.

Below, we present more systematic ways to produce quantifier-free formulae. In the meantime, let us see why we need modulo constraints in equivalent quantifier-free formulae; after all, $\exists\, \mathsf{z}\ \mathsf{x} = 2\mathsf{z}$ might admit another quantifier-free formula without modulo constraints. We explain below why it is impossible.

Let $\mathrm{AT}(\mathsf{x})$ be the set of atomic formulae of the form

$$a\mathsf{x} + b \leq a'\mathsf{x} + b',$$

where a, a', b and b' are natural numbers. Because x is interpreted by a natural number, every atomic formula $a\mathsf{x} + b \leq a'\mathsf{x} + b'$ of that form is equivalent to an atomic formula of the form either $\top$, $\bot$, $\mathsf{x} \leq k$ or $\mathsf{x} \geq k$ with $k \in \mathbb{N}$. For instance $3\mathsf{x} + 5 \leq \mathsf{x} + 8$ is equivalent to $\mathsf{x} \leq 1$. Now, one can show that for every formula ψ that is a Boolean combination of atomic formulae of the form either $\top$, $\bot$ or $\mathsf{x} \leq k$, $[\![\psi]\!]$ is a set of natural numbers equal to a *finite* union of intervals $\bigcup_i I_i$ such that each I_i is of the form either $[k_1, k_2]$ or $[k_1, +\infty[$ with $k_1, k_2 \in \mathbb{N}$. As a consequence, $\exists\, \mathsf{z}\ \mathsf{x} = 2\mathsf{z}$ cannot be equivalent to a formula ψ of the above form because $[\![\exists\, \mathsf{z}\ \mathsf{x} = 2\mathsf{z}]\!]$ is obviously not equal to a finite union of intervals of the form $\bigcup_i I_i$. Exercise 1.4 is dedicated to such a proof.

Now, let us provide the formal statement about quantifier elimination.

Theorem 1.14 (Quantifier elimination)**.** *For every formula φ, there exists an equivalent quantifier-free formula φ' such that φ' can be effectively built from φ. Moreover, $free(\varphi') \subseteq free(\varphi)$.*

In order to show that quantifiers can be eliminated, it is sufficient to establish quantifier elimination for formulae of the form $\exists\, \mathsf{x}\ \psi$ where ψ is a quantifier-free formula, i.e. a Boolean combination of atomic formulae of the form either $t \leq t'$ or $t \equiv_k t'$. We call this property (QE*). For example, let us consider the formula φ below:

$$\exists\, \mathsf{x}\ (\psi_0(\mathsf{x}) \wedge (\exists\, \mathsf{y}\ (\psi_1(\mathsf{x}, \mathsf{y}) \wedge \exists\, \mathsf{z}\ \psi_2(\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{z}'))))$$

where the ψ_i's are quantifier-free formulae. If $\exists\, \mathsf{z}\ \psi_2(\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{z}')$ is equivalent to the quantifier-free formula $\psi_2'(\mathsf{x}, \mathsf{y}, \mathsf{z}')$, then φ is equivalent to

$$\exists\, \mathsf{x}\ (\psi_0(\mathsf{x}) \wedge (\exists\, \mathsf{y}\ (\psi_1(\mathsf{x}, \mathsf{y}) \wedge \psi_2'(\mathsf{x}, \mathsf{y}, \mathsf{z}')))).$$

Indeed, the equivalence relation for formulae is a congruence for the logical connective (Boolean operators and quantifiers). Again, if $\exists\, \mathsf{y}\ (\psi_1(\mathsf{x}, \mathsf{y}) \wedge \psi_2'(\mathsf{x}, \mathsf{y}, \mathsf{z}')$ is equivalent to the quantifier-free formula $\psi_1'(\mathsf{x}, \mathsf{z}')$, then φ is equivalent to

$$\exists\, \mathsf{x}\ (\psi_0(\mathsf{x}) \wedge \psi_1'(\mathsf{x}, \mathsf{z}')).$$

Finally, if $\exists\, \mathsf{x}\, (\psi_0(\mathsf{x}) \wedge \psi_1'(\mathsf{x}, \mathsf{z}'))$ is equivalent to the quantifier-free formula $\psi_0'(\mathsf{z}')$, then φ is equivalent to it and $\psi_0'(\mathsf{z}')$ is quantifier-free.

So, given an arbitrary formula φ in FO($\mathbb{N}$) a way to compute an equivalent quantifier-free formula is the following. First, replace in φ every subformulae of the form $\forall\, \mathsf{x}\ \psi$ by $\neg\, \exists\, \mathsf{x}\, \neg\psi$ (in order to have only existential quantifications, possibly negated). Say, we get the formula φ'. If φ' is quantifier-free, we are done. Otherwise pick an innermost subformula of the form $\exists\, \mathsf{x}\, \chi$ where χ is quantifier-free and substitute it by an equivalent quantifier-free formula thanks to (QE*), leading to the new formula φ'' (equivalent to φ). The number of quantifiers in φ'' is strictly less than the number of quantifiers in φ'. By repeating this process (at most $|\varphi|$ times), we can compute a quantifier-free formula that is equivalent to the original formula φ.

1.3.2 Ingredients to eliminate quantifiers

Below, we explain how to eliminate quantifiers in formulae of the form used in (QE*). To do so, we start by a simple example.

Let $\varphi = \exists\, \mathsf{x}\ \psi$ be a formula such that ψ is a quantifier-free formula that is a Boolean combination of atomic formulae of one of the forms below: $\top$, $\bot$, $t \leq \mathsf{x}$, $t \leq t'$. We assume that the terms t, t' are of the extended form $(\sum_i a_i \mathsf{x}_i) + k$ where the a_i's and k belong to $\mathbb{Z}$ (and not only to $\mathbb{N}$) and x does not occur in the terms t or t'. This means that we have isolated x at one side of the inequalities. We have seen that it is harmless to use such extended terms since this is just a syntactic convention.

So, ψ is a formula built from the grammar below:

$$\chi ::= \top \ \mid \bot \mid\ t \leq \mathsf{x} \ \mid\ t \leq t' \ \mid\ \neg\chi \ \mid\ \chi \wedge \chi \tag{$\dagger$}$$

where t, t' are extended terms without occurrences of the variable x. Note that there is no formula of the form $t \geq \mathsf{x}$ since that is equivalent to $\neg(t + 1 \leq \mathsf{x})$.

Let us start by noticing that any variable valuation $\mathfrak{v} : \mathrm{VAR} \to \mathbb{N}$, can be generalized to extended terms such that

$$\mathfrak{v}((\sum_i a_i \mathsf{x}_i) + k) \stackrel{\text{def}}{=} (\sum_i a_i\ \mathfrak{v}(\mathsf{x}_i)) + k.$$

Terms can be interpreted by values in $\mathbb{Z}$ (but not the variables). Let T be the set of terms t occurring in ψ (excluding x) in atomic formulae of the form $t \leq \mathsf{x}$, and (possibly) augmented with 0.

Given a valuation $\mathfrak{v} : \mathrm{VAR} \to \mathbb{N}$, there is a term $t_{\text{left}} \in T$ such that $\mathfrak{v}(t_{\text{left}}) \leq \mathfrak{v}(\mathsf{x})$ and there is no term t in T such that $\mathfrak{v}(t_{\text{left}}) < \mathfrak{v}(t) \leq \mathfrak{v}(\mathsf{x})$. t_{left} is called the *closest left* term. Note that t_{left} always exists because 0 is in T but it may not be unique.

A key observation is that for any value $n \in [\mathfrak{v}(t_{\text{left}}), \mathfrak{v}(\mathsf{x})]$, $\mathfrak{v}$ and $\mathfrak{v}[\mathsf{x} \mapsto n]$ verifies exactly the same atomic formulae from ψ and therefore $\mathfrak{v} \models \psi$ iff $\mathfrak{v}[\mathsf{x} \mapsto$

$n] \models \psi$. Consequently, for the satisfaction of φ, we can always assume that x is equal to some term t with $t \in T$.

Below we write $\psi(\mathsf{x} \leftarrow t)$ to denote the formula obtained from ψ by replacing every occurrence of x by the term t. Now, one can show that φ is equivalent to the disjunction below:

$$\bigvee_{t \in T} \psi(\mathsf{x} \leftarrow t) \wedge t \geq 0.$$

Note that the disjunction can be computed in polynomial time in the size of φ. It is also worth observing that the existential quantification is replaced by a generalized disjunction, which is the way it should be conceptually.

Example 1.15. By way of example, let us consider the formula

$$\exists\, \mathsf{z}\, (\mathsf{x} < \mathsf{z} \wedge \mathsf{z} < \mathsf{y})$$

that has been already considered at the beginning of Section 1.3. First, we slightly update it to get an equivalent formula

$$\varphi = \exists\, \mathsf{z}\, (\mathsf{x} + 1 \leq \mathsf{z} \wedge \neg(\mathsf{y} \leq \mathsf{z})).$$

The set of terms T is equal to $\{\mathsf{x} + 1, \mathsf{y}, 0\}$. We get the following disjunction:

$$\begin{array}{c}
(\overbrace{\mathsf{x} + 1 \geq 0}^{\top} \wedge \overbrace{\mathsf{x} + 1 \leq \mathsf{x} + 1}^{\top} \wedge \neg(\mathsf{y} \leq \mathsf{x} + 1)) \vee \\
(\overbrace{\mathsf{y} \geq 0}^{\top} \wedge \mathsf{x} + 1 \leq \mathsf{y} \wedge \underbrace{\neg(\mathsf{y} \leq \mathsf{y})}_{\bot}) \vee \\
(\underbrace{0 \geq 0}_{\top} \wedge \underbrace{\mathsf{x} + 1 \leq 0}_{\bot} \wedge \neg(\mathsf{y} \leq 0))
\end{array}$$

which is equivalent to $\neg(\mathsf{y} \leq \mathsf{x} + 1)$. This formula is equivalent to $\mathsf{x} + 2 \leq \mathsf{y}$.

Let us extend a little bit what we have done with formulae of the form $\exists\, \mathsf{x}\, \psi$ when ψ respects (†). Indeed, quantifier-free formulae are much more general. For instance, inequalities may involve the terms $a\mathsf{x}$ with $a > 1$.

Let us consider $\varphi = \exists\, \mathsf{x}\, \psi$ when ψ follows the grammar below:

$$\chi ::= \top \mid \bot \mid t \leq a\mathsf{x} \mid t \leq t' \mid \neg\chi \mid \chi \wedge \chi \qquad (\dagger\dagger)$$

where t, t' are extended terms without occurrences of the variable x and $a \geq 1$.

Let l be the least common multiple (lcm) of all the coefficients occurring in front the variable x in ψ and let ψ' to denote the formula obtained from ψ by replacing every atomic formula of the form $t \leq a\mathsf{x}$ by $t \times \frac{l}{a} \leq l\mathsf{x}$. Multiplying by $\frac{l}{a}$ each side of an inequality provides an equivalent formula. Now, let ψ'' be the

formula obtained from ψ' by replacing every occurrence of $l\mathsf{x}$ by x. It is easy to show that φ and $\exists\, \mathsf{x}\ (\mathsf{x} \equiv_l 0) \wedge \psi''$ are equivalent. Hence, we consider $\varphi = \exists\, \mathsf{x}\ \psi$ when ψ follows the grammar below:

$$\chi ::= \top \mid \bot \mid t \equiv_k t' \mid \mathsf{x} \equiv_k t \mid t \leq \mathsf{x} \mid t \leq t' \mid \neg\chi \mid \chi \wedge \chi \qquad (\dagger\dagger\dagger)$$

where t, t' are extended terms without occurrences of the variable x, and $k \geq 1$. This clearly allows us to deal with the formula $(\mathsf{x} \equiv_l 0) \wedge \psi''$ described above. Quantifier-free formulae respecting $(\dagger\dagger\dagger)$ are almost of the general form except that modulo constraints or inequalities may involve the terms $a\mathsf{x}$ with $a > 1$, which is disallowed in $(\dagger\dagger\dagger)$. Such inequalities have been considered earlier, but let us consider modulo constraints of the form $a\mathsf{x} \equiv_k t$. If l is the lcm of all the coefficients occurring in front the variable x, $a\mathsf{x} \equiv_k t$ is equivalent to $l\mathsf{x} \equiv_{(k \times \frac{l}{a})} \frac{l}{a} t$ and therefore we can proceed as above.

Now, let us explain how to compute a quantifier-free formula equivalent to $\exists\, \mathsf{x}\ \psi$ when ψ belongs to $(\dagger\dagger\dagger)$; we know now that it entails (QE*) and therefore, quantifier elimination for FO($\mathbb{N}$) as well as its decidability.

Let l' be the least common multiple of all the coefficients occurring in modulo constraints in ψ. The developments below are obtained by slightly adapting what has been done for the case with $(\dagger)$ and by recalling that for all $n, n' \in \mathbb{N}$, $n \equiv_{l'} n'$ iff ($n \equiv_{k_1} n'$ and ... and $n \equiv_{k_\beta} n'$), which is a key property to deal with modulo constraints. Herein, l' is the lcm of the values $k_1, \ldots, k_\beta$.

Additionally, given a valuation $\mathfrak{v}$ defining a term $t_{\text{left}} \in T$ (see definition above), then for any value $n \in \{m \in [\mathfrak{v}(t_{\text{left}}), \mathfrak{v}(\mathsf{x})] : m \equiv_{l'} \mathfrak{v}(\mathsf{x})\}$, $\mathfrak{v}$ and $\mathfrak{v}[\mathsf{x} \mapsto n]$ verifies exactly the same atomic formulae from ψ and therefore $\mathfrak{v} \models \psi$ iff $\mathfrak{v}[\mathsf{x} \mapsto n] \models \psi$. Consequently, for the satisfaction of φ, we can always assume that x is equal to some term $t + j$ with $t \in T$ and $j \in [0, l' - 1]$.

Now, one can show that φ is equivalent to the disjunction below:

$$\bigvee_{t \in T, j \in [0, l'-1]} \psi(\mathsf{x} \leftarrow t + j) \wedge (t + j \geq 0)$$

Again, the very idea for eliminating the quantification over the variable x consists in considering a limited amount of values (obtained from terms that do not contain x) and to use a disjunction instead of a first-class quantification.

We wish to emphasize that this way to eliminate quantifiers is not completely optimal (even though the exponential blow-up is unavoidable in full generality) but it has allowed us to present the main ideas for the proof of (QE*).

Example 1.16. Let us consider the formula $\exists\, \mathsf{z}\ \mathsf{x} = 2\mathsf{z}$. In order to apply the above method, let us start by rewriting the formula in order to remove the equality predicate:

$$\varphi = \exists\, \mathsf{z}\ (\mathsf{x} \leq 2\mathsf{z}) \wedge (\neg(\mathsf{x} + 1 \leq 2\mathsf{z})).$$

Let us first eliminate the coefficient in front of z leading to the formula

$$\varphi = \exists\, \mathsf{z}\, (\mathsf{z} \equiv_2 0) \wedge (\mathsf{x} \leq \mathsf{z}) \wedge (\neg(\mathsf{x}+1 \leq \mathsf{z})).$$

The set of terms T is $\{0, \mathsf{x}, \mathsf{x}+1\}$. So $l' = 2$ and now we can introduce the generalized disjunction equivalent to φ:

$$\begin{array}{c}
[\overbrace{0 \geq 0}^{\top} \wedge (\overbrace{0 \equiv_2 0}^{\top}) \wedge (\mathsf{x} \leq 0) \wedge (\overbrace{\neg(\mathsf{x}+1 \leq 0)}^{\top})] \vee \\
[\underbrace{1 \geq 0}_{\top} \wedge (\underbrace{1 \equiv_2 0}_{\bot}) \wedge (\mathsf{x} \leq 1) \wedge (\neg \mathsf{x}+1 \leq 1)] \vee \\
[\overbrace{\mathsf{x} \geq 0}^{\top} \wedge (\mathsf{x} \equiv_2 0) \wedge (\overbrace{\mathsf{x} \leq \mathsf{x}}^{\top}) \wedge (\overbrace{\neg(\mathsf{x}+1 \leq \mathsf{x})}^{\top})] \vee \\
[\underbrace{\mathsf{x}+1 \geq 0}_{\top} \wedge (\mathsf{x}+1 \equiv_2 0) \wedge (\mathsf{x} \leq \mathsf{x}+1) \wedge (\underbrace{\neg \mathsf{x}+1 \leq \mathsf{x}+1}_{\bot})] \vee \\
[\underbrace{\mathsf{x}+1 \geq 0}_{\top} \wedge (\mathsf{x}+1 \equiv_2 0) \wedge (\mathsf{x} \leq \mathsf{x}+1) \wedge (\underbrace{\neg(\mathsf{x}+1 \leq \mathsf{x}+1)}_{\bot})] \vee \\
[\underbrace{\mathsf{x}+2 \geq 0}_{\top} \wedge (\mathsf{x}+2 \equiv_2 0) \wedge (\mathsf{x} \leq \mathsf{x}+2) \wedge (\underbrace{\neg(\mathsf{x}+1 \leq \mathsf{x}+2)}_{\bot})]
\end{array}$$

which is equivalent to $(\mathsf{x} \leq 0) \vee (\mathsf{x} \equiv_2 0)$ and therefore is equivalent to $\mathsf{x} \equiv_2 0$. This sounds as a lot of efforts for an obvious outcome but it has been done by following a systematic decision procedure that can be then automated.

1.4 Automata-based Approach for Presburger Arithmetic

In the previous section, we have seen that the satisfiability problem for Presburger arithmetic is decidable. In this section, we shall informally describe the decidability of satisfiability problem for Presburger arithmetic by translation into the non-emptiness problem for finite-state automata. This section is essentially borrowed from (Demri and Poitrenaud, 2011) following the developments in (Boudet and Comon, 1996). The use of automata for logical decision problems goes back to (Büchi, 1960a) and we shall provide below the approach by automatic structures developed in (Boudet and Comon, 1996) (see also (Wolper and Boigelot, 1995)). The seminal paper (Büchi, 1960b) describes how to use the automata-based approach to deal with Presburger arithmetic. Of course, other decision procedures exist for Presburger arithmetic: for instance, quantifier elimination method from (Reddy and Loveland, 1978) improves the method developped in (Cooper, 1972).

Before presenting the principles of the automata-based approach for Presburger arithmetic, let us mention that, in general, the automata-based approach consists in reducing logical problems into automata-based decision problems in

order to take advantage of known results from automata theory. Alternatively, this can be viewed as a means to transform declarative statements (typically formulae) into operational devices (typically automata with sometimes rudimentary computational power). The most standard target problems in automata used in this approach is the non-emptiness problem that checks whether an automaton admits at least one accepting computation. A pioneering work by Büchi (Büchi, 1960a) shows that Büchi automata are equivalent to formulae in monadic second-order logic (MSO) over $\langle \mathbb{N}, < \rangle$; models of a formula built over the second-order variables $P_1, \ldots, P_N$ are ω-sequences over the alphabet $\mathcal{P}(\{P_1, \ldots, P_N\})$. In full generality, the following are a few desirable properties of the approach.

- The reduction should be conceptually simple, apart from being semantically faithful.
- The computational complexity of the automata-based target problem should be well-characterised. In that way, a complexity upper bound is obtained to solve the source logical problem.
- Last but not least, the reduction should preferably allow the optimal complexity for the source logical problem to be obtained. (In the construction provided below, this last property is not satisfied.)

We have seen that each Presburger formula φ with $n \geq 1$ free variables defines a subset of $\mathbb{N}^n$, namely $[\![\varphi]\!] \subseteq \mathbb{N}^n$, that corresponds to the set of variable valuations that make φ true. For instance, $[\![\mathsf{x} = \mathsf{y} + \mathsf{z}]\!] = \{\langle k_1, k_2, k_3 \rangle \in \mathbb{N}^3 : k_1 = k_2 + k_3\}$. The automata-based approach for Presburger arithmetic consists of representing the tuples in $[\![\varphi]\!]$ by a regular language that can be effectively defined, for instance with the help of a finite-state automaton. In that way, satisfiability of φ, which is equivalent to the non-emptiness of $[\![\varphi]\!]$, becomes equivalent to the non-emptinesss of a finite-state automaton (which is an easy problem to solve once the automaton is built). In order to define regular languages, first we need to specify how natural numbers and tuples of natural numers are encoded by words over a finite alphabet. Numerous options are possible (see e.g. (Leroux, 2003; Klaedtke, 2004a)) and below we adopt a simple and standard encoding in which natural numbers are viewed as finite words over the alphabet $\{0, 1\}$ by using a binary representation in which the least significant bit is first.

We adopt a representation of natural numbers that is not unique, for instance the number five can be encoded by 101 or by 101000. Tuples of natural numbers of dimension n are represented by finite words over the alphabet $\{0, 1\}^n$ by using an equal length representation for each number (padding).

Typically, the pair $\left(\begin{smallmatrix} 5 \\ 8 \end{smallmatrix}\right)$ can be represented by the word

$$\left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right)\left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$$

over the alphabet $\{0, 1\}^2$. So, we introduce the map $\mathfrak{f} : \mathbb{N} \to \mathcal{P}(\{0, 1\}^*)$ such that $\mathfrak{f}(0) \stackrel{\text{def}}{=} 0^*$ and for $k > 0$, $\mathfrak{f}(k) \stackrel{\text{def}}{=} b_k \cdot 0^*$ where b_k is the shortest binary

representation of k (least significant bit first). The map $\mathfrak{f}$ is extended to subsets of $\mathbb{N}$ in the obvious way as well as to n-tuples of natural numbers with alphabet $\{0,1\}^n$ such that $\mathfrak{f}(\mathbf{x}) \subseteq \mathcal{P}((\{0,1\}^n)^*)$ with $\mathbf{x} \in \mathbb{N}^n$ and $\mathbf{b} \in \mathfrak{f}(\mathbf{x})$ iff for $i \in [1,n]$, the projection of $\mathbf{b}$ on the ith row belongs to $\mathfrak{f}(\mathbf{x}(i))$. The map $\mathfrak{f}$ is typically a state-encoding schema in the sense of (Legay, 2008).

Given a Presburger formula φ with $n \geq 1$ free variables and a finite-state automaton $\mathcal{A}$ over the alphabet $\{0,1\}^n$, we write $\varphi \approx \mathcal{A}$ whenever $\mathrm{L}(\mathcal{A}) = \mathfrak{f}([\![\varphi]\!])$.

Theorem 1.17. *(see e.g. (Boudet and Comon, 1996)) Given a Presburger formula φ, we can build a finite-state automaton $\mathcal{A}_\varphi$ such that $\varphi \approx \mathcal{A}_\varphi$.*

We also have $[\![\varphi]\!] \subseteq [\![\psi]\!]$ iff $\mathrm{L}(\mathcal{A}_\varphi) \subseteq \mathrm{L}(\mathcal{A}_\psi)$ (see e.g., (Legay, 2008, Theorem 3.22)).

The finite-state automaton $\mathcal{A}_\varphi$ can be built recursively over the structure of φ. For instance, conjunction is handled by the product construction, existential quantifier is handled by projection, negation is handled by the complement construction, see details below. Nevertheless, a crude complexity analysis of the construction of $\mathcal{A}_\varphi$ reveals a non-elementary worst-case complexity. Indeed, for every negation, a complementation needs to be operated. However, developments related to the optimal size of automata can be found in (Klaedtke, 2004b).

The recursive definition is based on the following properties. Let φ and ψ be Presburger formulae with free variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$.

conjunction If $\varphi \approx \mathcal{A}$ and $\psi \approx \mathcal{B}$, then $\varphi \wedge \psi \approx \mathcal{A} \cap \mathcal{B}$ where $\cap$ is the product construction computing intersection;

negation If $\varphi \approx \mathcal{A}$, then $\neg\varphi \approx \overline{\mathcal{A}}$ where $\bar{\cdot}$ performs complementation, which may cause an exponential blow-up;

quantification If $\varphi \approx \mathcal{A}$, then $\exists\, \mathsf{x}_n\ \varphi \approx \mathcal{A}'$ where $\mathcal{A}'$ is built over the alphabet $\{0,1\}^{n-1}$ by forgetting the nth component. Typically, $q \xrightarrow{\mathbf{b}} q'$ in $\mathcal{A}'$ whenever there is a transition $q \xrightarrow{\mathbf{b}'} q'$ in $\mathcal{A}$ such that $\mathbf{b}$ and $\mathbf{b}'$ agree on the $n-1$ first bit values.

In the above construction, we assumed that φ and ψ share the same set of free variables, which does not always hold true for arbitrary formulae. If it is not the case, $\varphi \approx \mathcal{A}$ and $\psi \approx \mathcal{B}$, then we perform an operation that consists of adding dummy bits. For instance, suppose that φ contains the free variables $\mathsf{x}_1, \ldots, \mathsf{x}_n$. We can build the automaton $\mathcal{A}'$ over the alphabet $\{0,1\}^{n+1}$ obtained by adding the $(n+1)$th component. Typically, $q \xrightarrow{\mathbf{b}} q'$ in $\mathcal{A}'$ whenever there is a transition $q \xrightarrow{\mathbf{b}'} q'$ in $\mathcal{A}$ such that $\mathbf{b}$ and $\mathbf{b}'$ agree on the n first bit values. It remains to deal with atomic formulae to achieve the inductive building of the automaton.

The proof of theorem 1.17 is clearly based on the above constructions but we need to complete the argument in order to deal with atomic formulae. Without

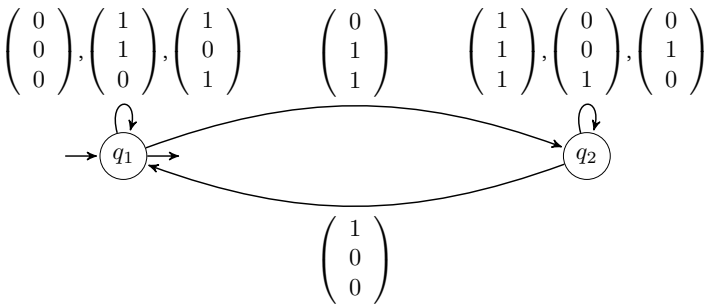

Figure 1.2: Automaton for $\mathsf{x}_1 = \mathsf{x}_2 + \mathsf{x}_3$

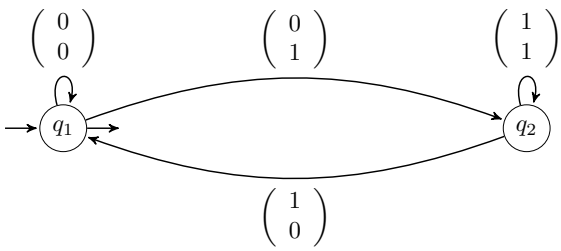

Figure 1.3: Automaton for $\mathsf{x}_1 = \mathsf{x}_2 + \mathsf{x}_2$

---

any loss of generality, we can restrict ourselves to equalities of the form $\mathsf{x} = \mathsf{y} + \mathsf{z}$ (at the cost of introducing new variables in order to deal with sums made of more than two variables). Such a restriction is only helpful to simplify the presentation of the method but it makes sense to consider the full language with linear constraints in order to optimize the reduction to automata, see e.g. (Boigelot and Wolper, 2002; Boudet and Comon, 1996). The variables in $\mathsf{x} = \mathsf{y} + \mathsf{z}$ are not necessarily distinct.

The automaton for $\mathsf{x}_1 = \mathsf{x}_2 + \mathsf{x}_3$ is described in Figure 1.2 where q_1 is the initial state as well as the final state. The state q_1 represents a carry-over of 0 whereas the state q_2 represents a carry-over of 1. We can check that $(\mathsf{x}_1 = \mathsf{x}_2 + \mathsf{x}_3) \approx \mathcal{A}$. The Figure 1.2 describes the automaton for $\mathsf{x}_1 = \mathsf{x}_2 + \mathsf{x}_2$.

By projection, the encoding for $\exists\, \mathsf{x}_2\ (\mathsf{x}_1 = \mathsf{x}_2 + \mathsf{x}_2)$ is presented in Figure 1.4. Note that the automaton recognizes preciely the encodings of even numbers.

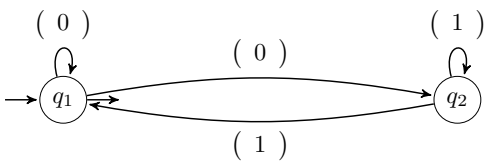

Figure 1.4: Encoding for $\exists\, \mathsf{x}_2\ (\mathsf{x}_1 = \mathsf{x}_2 + \mathsf{x}_2)$.

The automata-based approach for Presburger arithmetic can be extended to richer theories such as $\langle \mathbb{R}, \mathbb{Z}, +, \leq \rangle$, see e.g. (Boigelot and Wolper, 2002), or can be refined by providing other reductions, see e.g. (Leroux, 2003; Klaedtke, 2004b; Schuele and Schneider, 2007). An overview of automata-based decision procedures for Presburger arithmetic and related formalisms can be found in (Klaedtke, 2004b).

In (Durand-Gasselin and Habermehl, 2010), it is also shown that the translation from formulae in FO($\mathbb{N}$) to finite-state automata can be done in triple exponential time, when the reduction from (Boudet and Comon, 1996), that is simpler conceptually, would lead to a non-elementary bound since each negation may cause an exponential blow-up. Automata constructions for fragments of FO($\mathbb{N}$) are also studied in (Durand-Gasselin and Habermehl, 2010).

1.5 Semilinear Sets

In this section, we present a standard characterization for the Presburger sets, i.e. for the sets that are equal to $[\![\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_d)]\!]$ for some formula in FO($\mathbb{N}$) with free variables included in $\mathsf{x}_1, \ldots, \mathsf{x}_d$, $d \geq 1$. Before presenting the correspondence, let us deal with the one-variable case, that will be then extended.

1.5.1 Formulae with at most one free variable

Formulae with at most one free variable can use more than one variable but those variables need to be quantified and therefore to be in the scope of a quantifier. By way of example, let us consider the formula $\varphi(\mathsf{x})$ below:

$$(\mathsf{x} \neq 1 \wedge \mathsf{x} \neq 2) \wedge (\mathsf{x} = 0 \vee (\mathsf{x} \geq 3 \wedge \exists\, \mathsf{y}\, (\mathsf{x} = 3 + 2\mathsf{y}))).$$

It is easy to show that $[\![\varphi(\mathsf{x})]\!] = \{0\} \cup \{3 + 2n : n \geq 0\}$ and therefore after the value 3, every two values belongs to the set $[\![\varphi(\mathsf{x})]\!]$. Such a regularity can be slightly generalized leading to the definition below.

Definition 1.18 (Ultimately periodic set)**.** A set $X \subseteq \mathbb{N}$ is *ultimately periodic* $\overset{\text{def}}{\Leftrightarrow}$ there exist $N \geq 0$, and $P \geq 1$ such that for all $n \geq N$, we have $n \in X$ iff $n + P \in X$.

The set of even numbers is ultimately periodic (with $N = 0$ for instance and $P = 2$) as well as the set of odd numbers (with $N = 1$ and $P = 2$ too). More generally, one can easily show that $[\![\mathsf{x} \equiv_k k']\!]$ is ultimately periodic (with $N = 0$ and $P = k$).

Ultimately periodic sets satisfy nice closure properties thanks to their regularity.

Proposition 1.19. *Ultimately periodic sets are closed under union, intersection and complementation.*

For instance, suppose that X is ultimately periodic and let $\overline{X} = \mathbb{N} \setminus X$. The statements below are equivalent for $n \geq N$:

- $n \in \overline{X}$,
- $n \notin X$ (by definition of $\overline{X}$),
- $n + P \notin X$ (X is ultimately periodic with parameters N and P),
- $n + P \in \overline{X}$ (by definition of $\overline{X}$).

So $\overline{X}$ is ultimately periodic too and the same parameters N and P can be used. The proof for union and intersection is left as an exercise.

Moreover, ultimately periodic sets can be easily defined by formulae with at most one free variable as shown below. Indeed, let X be such a set defined with the help of N and P (that are not unique). The formula φ_X below verifies that $[\![\varphi_X]\!] = X$.

$$(\bigwedge_{k \in [0,N-1] \setminus X} \mathsf{x} \neq k) \wedge [(\bigvee_{k \in [0,N-1] \cap X} \mathsf{x} = k) \vee$$
$$((\mathsf{x} \geq N) \wedge (\exists\, \mathsf{y} \bigvee_{k \in [N,N+P-1] \cap X} (\mathsf{x} = k + P\mathsf{y})))]$$

The converse is also true as stated below.

Proposition 1.20. *For every formula $\varphi(\mathsf{x})$ in $\mathrm{FO}(\mathbb{N})$ with a unique free variable x, $[\![\varphi]\!]$ is an ultimately periodic set.*

Proof. Let $\varphi(\mathsf{x})$ be a formula with a unique free variable and φ' be an equivalent quantifier-free formula (by Theorem 1.14). φ' can be shown equivalent to a Boolean combination of atomic formulae of one of the forms below: $\top$, $\bot$, $\mathsf{x} \leq k$, $\mathsf{x} \equiv_k k'$. Each atomic formula defines an ultimately periodic set and ultimately periodic sets are closed under union, intersection and complementation. So $[\![\varphi']\!]$ is ultimately periodic and since φ and φ' are equivalent formulae, $[\![\varphi]\!]$ is ultimately periodic too. □

What has been done with ultimately periodic sets can be generalized with semilinearity sets and semilinearity sets will also satisfy nice closure properties (a must to capture logical connectives from $\mathrm{FO}(\mathbb{N})$). This is the subject of the next section.

1.5.2 Semilinear sets

A *linear set* X (of dimention $d \geq 1$) is defined as a subset of $\mathbb{N}^d$ for which there exists a *basis* $\mathbf{b} \in \mathbb{N}^d$ and a finite set of *periods* $\mathfrak{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_m\} \subseteq \mathbb{N}^d$ such that

$$X = \{\mathbf{b} + \sum_{i=1}^{m} \lambda_i \mathbf{p}_i : \lambda_1, \ldots, \lambda_m \in \mathbb{N}\}.$$

For example, the set of even numbers $\{0 + i \times 2 : i \in \mathbb{N}\}$ is a linear subset of $\mathbb{N}$ (dimension 1). Similarly, $\{1 + i \times 2 : i \in \mathbb{N}\}$ is a linear set with $\mathbf{b} = 1$ and with unique period 2. A *semilinear set* is defined as a finite union of linear sets. Each semilinear set can be represented by a finite set of pairs of the form $\langle \mathbf{b}, \mathfrak{P} \rangle$. Here is a linear set of dimension 2:

$$\left\{ \begin{pmatrix} 3 \\ 4 \end{pmatrix} + i \times \begin{pmatrix} 2 \\ 5 \end{pmatrix} + j \times \begin{pmatrix} 4 \\ 7 \end{pmatrix} : i, j \in \mathbb{N} \right\}.$$

Remark 1.21. Semilinear sets generalize the notion of ultimately periodic sets in dimension 1 since every ultimately periodic set X is a semilinear set of dimension 1. Indeed, suppose that X is characterized by the parameters N and P. The equality below is easy to show:

$$X = (\bigcup_{n \in [0, N-1] \cap X} \{n\}) \cup (\bigcup_{n \in [N, N+P-1] \cap X} \{n + \lambda P : \lambda \in \mathbb{N}\})$$

Note that each singleton set $\{n\}$ is a linear set (with no period, or equivalently with all periods equal to zero) as well as each set $\{n + \lambda P : \lambda \in \mathbb{N}\}$ (with basis n and period P).

Semilinear sets satisfy nice closure properties.

Theorem 1.22. *The class of semilinear sets are effectively closed under union, intersection and complementation.*

Closure by union is immediate from the definition of semilinear sets. The class of semilinear sets happen to be more interesting since it corresponds exactly to the Presburger sets (see Theorem 1.23). Observe that if X_1 and X_2 are semilinear subsets of $\mathbb{N}^d$ such that $X_1 = [\![\varphi_1]\!]$ and $X_2 = [\![\varphi_2]\!]$, then $X_1 \cap X_2 = [\![\varphi_1 \wedge \psi_2]\!]$ and $\mathbb{N}^d \setminus X_1 = [\![\neg\varphi_1]\!]$. For example, let us consider the following semilinear set:

$$\left\{ \begin{pmatrix} 3 \\ 4 \end{pmatrix} + i \times \begin{pmatrix} 2 \\ 5 \end{pmatrix} + j \times \begin{pmatrix} 4 \\ 7 \end{pmatrix} \mid i, j \in \mathbb{N} \right\}.$$

The Presburger formula below can define it (and its principle can be generalized to any linear set):

$$\exists\, \mathsf{y}, \mathsf{y}' \; (\mathsf{x}_1 = 3 + 2\mathsf{y} + 4\mathsf{y}' \wedge \mathsf{x}_2 = 4 + 5\mathsf{y} + 7\mathsf{y}').$$

Theorem 1.23. *Semilinear sets coincide with Presburger sets, i.e.*

(I) *for every formula $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_d)$ with $d \geq 1$, $[\![\varphi]\!]$ is a semilinear set and its representation can be effectively computed,*

(II) *for every semilinear set $X \subseteq \mathbb{N}^d$, there is a formula φ (and it can be computed) such that $X = [\![\varphi]\!]$.*

Since checking whether a linear set given as a basis and a finite set of periods is empty can be easily checked, Theorem 1.23 entails the decidability of the satisfiability problem for FO($\mathbb{N}$).

Proof. (sketch) (I) Let φ be a formula with free variables $\mathsf{x}_1, \ldots, \mathsf{x}_d$. By Theorem 1.14, there is a quantifier-free formula ψ equivalent to φ. By Theorem 1.22, the set of semilinear sets is closed union, intersection and complementation. So, it remains to show that atomic formulae define semilinear sets too, which is left as an exercise. (II) Every linear set can be easily shown to be a Presburger set by generalizing the above construction. Disjunction in FO($\mathbb{N}$) allows to deal with finite union of linear sets. □

Example 1.24. Let us show that the set $Y = \{n^2 : n \in \mathbb{N}\}$ is not a Presburger set by showing that Y is not semilinear. The proof is *ad absurdum*. Since Y is infinite, there are $\mathbf{b} \geq 0$ and $\mathbf{p}_1, \ldots, \mathbf{p}_m > 0$ ($m \geq 1$) such that $Z \stackrel{\text{def}}{=} \{\mathbf{b} + \sum_{i=1}^{m} \lambda_i \mathbf{p}_i : \lambda_1, \ldots, \lambda_m \in \mathbb{N}\} \subseteq Y$ and Z is infinite. Let $N \in \mathbb{N}$ be such that $N^2 \in Z$ and $(2N+1) > \mathbf{p}_1$. The value N always exists since Z is infinite. Since Z is a linear set, we also have $(N^2 + \mathbf{p}_1) \in Z$. However $(N+1)^2 - N^2 = (2N+1) > \mathbf{p}_1$. Hence $N^2 < N^2 + \mathbf{p}_1 < (N+1)^2$, which leads to a contradiction since there is no square value strictly before N^2 and $(N+1)^2$.

Example 1.25. Let us show that $X = \{2^i : i \in \mathbb{N}\}$ is not semilinear. The proof is also *ad absurdum*. Suppose that X is semilinear. Since X is an infinite set, there exist a basis $\mathbf{b} \in \mathbb{N}$ and period(s) $\mathbf{p}_1, \ldots, \mathbf{p}_m \in (\mathbb{N} \smallsetminus \{0\})$ ($m \geq 1$) such that $Y = \{\mathbf{b} + \sum_{i=1}^{m} \lambda_i \mathbf{p}_i : \lambda_1, \ldots, \lambda_m \in \mathbb{N}\} \subseteq X$ and Y is infinite. There exists $2^\alpha \in Y$ such that $\mathbf{p}_1 < 2^\alpha$. By definition of Y, we have $2^\alpha + \mathbf{p}_1 \in Y$ but $2^\alpha < 2^\alpha + \mathbf{p}_1 < 2^{\alpha+1}$, which leads to a contradiction.

Example 1.26. Let us consider the VASS in Figure 1.5 with six counters and two control states. Actually, the VASS weakly computes multiplication and one can establish the following result.

$$\left\{ \begin{pmatrix} a \\ b \\ f \end{pmatrix} \in \mathbb{N}^3 \mid \exists \begin{pmatrix} c \\ d \\ e \end{pmatrix} \in \mathbb{N}^3,\ \langle q_0, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rangle \xrightarrow{*} \langle q_1, \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} \rangle \right\} =$$

$$\left\{ \begin{pmatrix} n \\ m \\ p \end{pmatrix} \in \mathbb{N}^3 : p \leq n \times m \right\}.$$

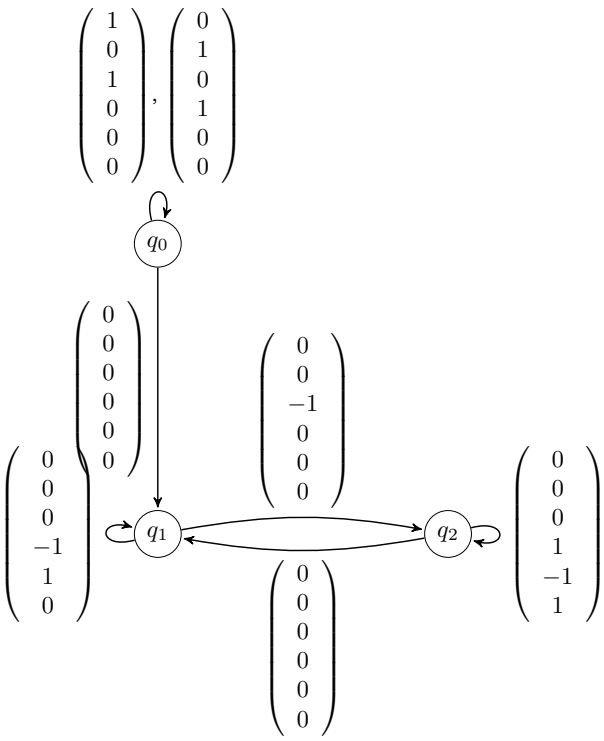

Figure 1.5: Hopcroft & Pansiot's VASS for weak multiplication

---

Let us show that there is no formula $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_6)$ such that

$$[\![\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_6)]\!] = \{ \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} \mid \langle q_0, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \rangle \xrightarrow{*} \langle q_1, \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} \rangle \}$$

The proof is simple and *ad absurdum*. Suppose that $\varphi(\overline{\mathsf{x}})$ exists. Then, the formula $\psi(\mathsf{x})$ defined below verifies $[\![\psi(\mathsf{x})]\!] = \{n^2 \mid n \in \mathbb{N}\}$

$$\psi(\mathsf{x}) \stackrel{\text{def}}{=} \exists\, \mathsf{x}_1, \ldots, \mathsf{x}_5\ \varphi(\mathsf{x}_1, \ldots, \mathsf{x}_5, \mathsf{x}) \wedge \mathsf{x}_1 = \mathsf{x}_2 \wedge$$

$$\forall\, \mathsf{x}'\ (\mathsf{x}' > \mathsf{x}) \Rightarrow \neg \exists\, \mathsf{x}_3, \mathsf{x}_4, \mathsf{x}_5\ \varphi(\mathsf{x}_1, \ldots, \mathsf{x}_5, \mathsf{x}')$$

which leads to a contradiction since $\{n^2 \mid n \in \mathbb{N}\}$ is not a Presburger set.

1.6 Application: Parikh Image of Regular Languages

In this section, we show that the commutative image of a regular language can be characterized by a formula in FO($\mathbb{N}$).

1.6.1 Parikh image

Let $\Sigma = \{a_1, \ldots, a_k\}$ equipped with an arbitrary linear ordering of the letters, say $a_1 < \cdots < a_k$. Given a word $u \in \Sigma^*$, its *Parikh image* is defined as a tuple $\Pi(u) \in \mathbb{N}^k$ such that for every $i \in [1, k]$, $\Pi(u)(i)$ is the number of occurrences of the letter a_i in the word u. For instance, the Parikh image of the word $abaab$ under the ordering $a < b$ is the tuple $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$. Naturally, the *Parikh image* of the language $\mathrm{L} \subseteq \Sigma^*$ is the set $\{\Pi(u) \in \mathbb{N}^k : u \in \mathrm{L}\}$. Parikh's remarkable result states that the Parikh image of any context-free language is semilinear and that its representation is effectively computable from pushdown automata or from a context-free grammar. Below, we present a proof for this result for the class of regular languages by introducing the class of bounded and regular languages.

1.6.2 Semilinearity by bounded languages

The goal of this section is to show that the Parikh image of regular languages is definable by a formula in $\mathrm{FO}(\mathbb{N})$. To do so, we show a nice result about the fact that every regular language contains a bounded and regular language with the same Parikh image. First, let us recall what are bounded languages.

Let Σ be a finite alphabet. A language $\mathrm{L} \subseteq \Sigma^*$ is called *bounded* $\overset{\text{def}}{\Leftrightarrow}$ there exists a finite sequence $u_1, \ldots, u_n$ of words in Σ^* such that $\mathrm{L} \subseteq u_1^* \cdots u_n^*$. Since regular languages are recognized by deterministic finite-state automata, a language $\mathrm{L} \subseteq \Sigma^*$ is bounded and regular if, and only if, L is a finite union of languages of the form $u_0 v_1^* u_1 \cdots v_k^* u_k$ where $u_0, v_1, u_1, \ldots, v_k, u_k$ are words in Σ^*.

The Parikh images of bounded and regular languages can be easily shown semilinear and therefore they are Presburger sets.

Lemma 1.27. *For every bounded and regular language* L, $\Pi(\mathrm{L})$ *is semilinear.*

Indeed, the Parikh image of any language $u_0 v_1^* u_1 \cdots v_k^* u_k$ is equal to the linear set $\{\mathbf{b} + \lambda_1 \mathbf{p}_1 + \cdots \lambda_k \mathbf{p}_k : \lambda_1, \ldots, \lambda_k \in \mathbb{N}\}$ with

- $\mathbf{b} = \Pi(u_0) + \cdots + \Pi(u_k)$,
- $\mathbf{p}_i = \Pi(v_i)$ for every $i \in [1, k]$.

We recall that a *finite-state automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, F \rangle$ such that

- Σ is a finite *alphabet*,
- Q is a finite set of *states*,
- $Q_0 \subseteq Q$ is the set of *initial* states,
- the *transition relation* δ is a subset of $Q \times \Sigma \times Q$,
- $F \subseteq Q$ is a set of *accepting* states.

A *run* ρ of $\mathcal{A}$ is a sequence $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ldots$ such that for every $i \geq 0$, $\langle q_i, a_i, q_{i+1} \rangle \in \delta$ (also written $q_i \xrightarrow{a_i} q_{i+1}$). The finite run $\rho = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ldots \xrightarrow{a_{n-1}} q_n$ is *accepting* if $q_0 \in Q_0$ and $q_n \in F$. The *label* of ρ, written $lab(\rho)$, is the finite word $u = a_0 a_1 \cdots a_{n-1}$. The automaton $\mathcal{A}$ *accepts* the language $\mathrm{Lan}(\mathcal{A})$ of finite words $u \in \Sigma^*$ such that there exists an accepting run of $\mathcal{A}$ on the word u, i.e., with label u. Regular languages are those definable by finite-state automata.

Theorem 1.28. *For every regular language* L*, there is a bounded and regular language* L′ *such that* $\mathrm{L}' \subseteq \mathrm{L}$ *and* $\Pi(\mathrm{L}') = \Pi(\mathrm{L})$.

This is probably more than what we need to show that the Parikh image of regular languages is semilinear but the proof technique for the above theorem will be reused.

Proof. Let $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, F \rangle$ be a finite-state automaton accepting L. Without any loss of generality, we can assume that $Q_0 \cap F \neq \emptyset$ (otherwise add the empty string ε to the bounded language). A *path* π is a finite sequence of transitions from δ corresponding to a path in the control graph of $\mathcal{A}$. So, paths are essentially runs but we wish to emphasize here that we are only taking care of the graph structure of $\mathcal{A}$. We write $lab(\pi)$ to denote its label as a word of Σ^*. A *simple loop* sl is a non-empty path that starts and ends by the same state and these are the only states that are repeated in sl. We say that sl *loops* on its first state (equal to its last state). The number of simple loops is therefore bounded by $\mathrm{card}(\delta)^{\mathrm{card}(Q)}$. We assume an arbitrary total linear ordering $\prec$ on simple loops. An *extended path* **P** is an expression of the form below:

$$\pi_0 \; S_1 \; \pi_1 \; \cdots \; S_\alpha \; \pi_\alpha$$

where the S_i's are non-empty sets of simple loops, the π_i's are non-empty paths and

1. if S occurs just before a path π, then all the simple loops in S loops on the first state of π,
2. similarly, if S occurs just after a path π, then all the simple loops in S loops on the last state of π.

An extended path generalizes the notion of path in which simple loops in the S_i's can be visited an arbitrary number of times but respecting the arbitrary linear ordering on simple loops. Note also that alternative definitions are possible, for example by allowing that S_i is empty and by allowing that an extended path starts or ends by a set of simple loops. However, the definition below will be convenient for the forthcoming developments.

Given an extended path **P**, we introduce a few more notions.

- The *skeleton* of $\mathbf{P}$ is the path $\pi_0 \cdots \pi_\alpha$.
- Given a set of simple loops $S = \{sl_1, \ldots, sl_m\}$ with $sl_1 \prec \cdots \prec sl_m$ (remember, we fixed an arbirary ordering on simple loops), we write $e(S)$ to denote the regular expression

$$lab(sl_1)^+ \cdots lab(sl_m)^+$$

  So, each simple loop is taken at least once. Indeed, we want to make explicit that each simple loop sl is used in S, otherwise it is always possible to exclude sl from S, leading to another legitimate extended path. We write $e(\mathbf{P})$ to denote the regular expression defined as follows (if sl is not the only element in S):

$$lab(\pi_0) \cdot e(S_1) \cdots e(S_\alpha) \cdot lab(\pi_\alpha).$$

We write Lan(e) to denote the language defined by the regular expression e. Note that when the first state occuring in the skeleton of $\mathbf{P}$ is in Q_0 and the last state is in F, then Lan$(e(\mathbf{P})) \subseteq$ Lan$(\mathcal{A})$. In that case, we say that the extended path is *accepting*. One can observe that Lan$(e(\mathbf{P}))$ is a bounded and regular language for any extended path $\mathbf{P}$. For the sake of simplicity, we write Lan$(\mathbf{P})$ instead of Lan$(e(\mathbf{P}))$.

An alternative notion of extended path would be to define them as sequences of the form $\mathbf{P} = \pi_0, sl_1, \pi_1, \ldots, sl_\alpha, \pi_\alpha$, where the π_i's are (possibly empty) paths and the sl_i's are simple loops and to define $e(\mathbf{P})$ as $lab(\pi_0) \cdot lab(sl_1)^+ \cdot lab(\pi_1) \cdots lab(sl_\alpha)^+ \cdot lab(\pi_\alpha)$. The developments below, could be adapted to this slight variant. The current definition of extended paths with sets S of simple loops allows to restrict ourselves to sequences of the form

$$\pi_0, sl_1^1, \ldots, sl_{m_1}^1, \pi_1, \ldots, sl_\alpha^1, \ldots, sl_{m_\alpha}^1, \pi_\alpha$$

with additional constraints, for instance sl_i^j and $sl_i^{j'}$ loop on the same control state and $sl_i^j \prec sl_i^{j'}$ if $j < j'$. Note also that the linear ordering $\prec$ is arbitrary.

A *small* extended path is an extended path such that

1. π_0 and π_α have at most $2 \times$ card(Q) transitions,
2. $\pi_1, \ldots, \pi_{\alpha-1}$ have at most card(Q) transitions,
3. for each state $q \in Q$, there is at most one set S containing simple loops on q.

So, the length of the skeleton is bounded by card$(Q)(3 +$ card$(Q))$. Note that the set of small extended paths is finite, even though its cardinal can be exponential in the size of $\mathcal{A}$. We also consider degenerated small extended paths made of paths of length at most $3 \times$ card(Q). Usually, this case is omitted in the proofs since it can be easily derived from the non-degenerated case.

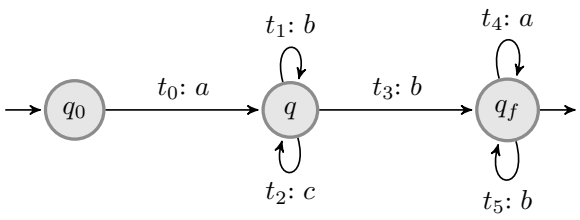

Figure 1.6: A simple finite-state automaton.

Example 1.29. Figure 1.6 presents a simple finite-state automaton; the edges are labelled by the name of the transition and by the letter. Here is an example of small extended path $\mathbf{P}$:

$$t_0 \cdot t_1 \cdot \{t_1, t_2\} \cdot t_3 \cdot \{t_4, t_5\} \cdot t_4 \cdot t_5 \cdot t_5$$

Note that $e(\mathbf{P})$ is equal to the regular expression below assuming that $t_1 \prec t_2$ and $t_5 \prec t_4$:

$$a \cdot b \cdot b^+ \cdot c^+ \cdot b \cdot b^+ \cdot a^+ \cdot a \cdot b \cdot b$$

Let X be the set of accepting and small extended paths from $\mathcal{A}$. Below we show that

$$\Pi(\bigcup_{\mathbf{P} \in X} \mathrm{Lan}(\mathbf{P})) = \Pi(\mathrm{Lan}(\mathcal{A})).$$

We already observed that $(\bigcup_{\mathbf{P} \in X} \mathrm{Lan}(\mathbf{P})) \subseteq \mathrm{Lan}(\mathcal{A})$ and therefore

$$\Pi(\bigcup_{\mathbf{P} \in X} \mathrm{Lan}(\mathbf{P})) \subseteq \Pi(\mathrm{Lan}(\mathcal{A})).$$

Let ρ be an accepting run of $\mathcal{A}$ and therefore $lab(\rho) \in \mathrm{Lan}(\mathcal{A})$ and $\Pi(lab(\rho)) \in \Pi(\mathrm{Lan}(\mathcal{A}))$. We shall build a small and accepting extended path $\mathbf{P}$ such that $\Pi(lab(\rho)) \in \Pi(\mathrm{Lan}(\mathbf{P}))$. To do so, we define a sequence of accepting extended paths $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_\beta$ such that

- all the $\mathbf{P}_i$'s are accepting extended paths,
- $\mathbf{P}_0$ is equal to ρ viewed as an extended path,
- $\mathbf{P}_\beta$ is a small and accepting extended path,
- $\mathbf{P}_{i+1}$ is obtained from $\mathbf{P}_i$ by removing a simple loop on q and possibly adding it to a set of simple loops S already in $\mathbf{P}_i$ or by creating one if none, and $\Pi(\mathrm{Lan}(\mathbf{P}_i)) \subseteq \Pi(\mathrm{Lan}(\mathbf{P}_{i+1}))$.

So, at the end of this process, $\Pi(lab(\rho)) \in \Pi(\mathrm{Lan}(\mathbf{P}_\beta))$ and $\Pi(\mathrm{Lan}(\mathbf{P}_\beta)) \subseteq \Pi(\mathrm{Lan}(\mathcal{A}))$.

It remains to explain how to build $\mathbf{P}_{i+1}$ from $\mathbf{P}_i$. We assume that $\mathbf{P}_i$ has the form below

$$\pi_0\ S_1\ \pi_1\ \cdots\ S_\alpha\ \pi_\alpha$$

where

(a) $\alpha \leq \text{card}(Q)$,

(b) each path in $\pi_1, \ldots, \pi_{\alpha-1}$ have length less than $\text{card}(Q)$,

(c) each state has at most one S_i containing simple loops on it.

Obviously, $\mathbf{P}_0$ verifies these conditions. $\mathbf{P}_{i+1}$ will satisfy the same condition except that we require that the length of the final path of $\mathbf{P}_{i+1}$ strictly decreases. Now, let us define $\mathbf{P}_{i+1}$ from $\mathbf{P}_i$.

Case 1: $\mathbf{P}_i$ is a small extended path. We are done and $\mathbf{P}_i$ is the final extended path of the sequence.

Case 2: $\pi_\alpha = \pi \cdot sl \cdot \pi'$ where sl is a simple loop on q, $\pi\pi' \neq \varepsilon$ and S_γ already contains simple loops on q ($\gamma \leq \alpha$). Then, $\mathbf{P}_{i+1}$ is equal to the extended path below:

$$\pi_0\ \cdots\ S_{\gamma-1}\ \pi_{\gamma-1}\ (S_\gamma \cup \{sl\})\ \cdots\ \pi_{\alpha-1}\ S_\alpha\ (\pi\pi')$$

Case 3: $\pi_\alpha = \pi \cdot sl \cdot \pi'$ where sl is a simple loop on q and the first one occurring in $\pi \cdot sl$, $\pi\pi' \neq \varepsilon$ and no S_γ already contains simple loops on q. Then, $\mathbf{P}_{i+1}$ is equal to the extended path below:

$$\pi_0\ \cdots\ S_\alpha\ \pi\ \{sl\}\ \pi'$$

In that case, we create a new (singleton) set of simple loops.

It remains to show the following properties, which is left as an exercise:

1. $\Pi(\text{Lan}(\mathbf{P}_i)) \subseteq \Pi(\text{Lan}(\mathbf{P}_{i+1}))$.
2. $\mathbf{P}_{i+1}$ satisfies the conditions (a), (b) and (c).
3. $\text{Lan}(\mathbf{P}_{i+1}) \subseteq \text{Lan}(\mathcal{A})$. □

By way of example, suppose that $\mathbf{P}_0$ is equal to the (extended) path below from the finite-state automaton defined in Figure 1.6:

$$t_0 \cdot (t_1)^7 \cdot (t_2)^7 (t_1)^8 \cdot t_3 \cdot (t_4)^7 \cdot (t_5)^7 \cdot (t_4)^8$$

We obtain the following extended path with the procedure described above:

- $\mathbf{P}_{22} = t_0 \cdot \{t_1, t_2\} \cdot t_3 \cdot (t_4)^7 \cdot (t_5)^7 \cdot (t_4)^8$,
- $\mathbf{P}_{38} = t_0 \cdot \{t_1, t_2\} \cdot t_3 \cdot \{t_4, t_5\} \cdot (t_4)^6$ and $\mathbf{P}_{38}$ is a small extended path.

Corollary 1.30. *Let $\mathcal{A}$ be a finite-state automaton over the alphabet Σ, equipped with a linear ordering of the letters, say with k letters. Then, one can compute a formula $\varphi_{\mathcal{A}}(\mathsf{x}_1, \ldots, \mathsf{x}_k)$ in* $\mathrm{FO}(\mathbb{N})$ *such that* $\Pi(\mathrm{Lan}(\mathcal{A})) = [\![\varphi_{\mathcal{A}}]\!]$.

Indeed, by Theorem 1.28, $\mathrm{Lan}(\mathcal{A})$ includes a bounded and regular language L with the same Parikh image. Note that L can be computed for instance by enumerating the regular expressions obtained from small and accepting extended paths and then check inclusion with $\mathrm{Lan}(\mathcal{A})$. Then, we compute a disjunction made of the formulae obtained for each bounded and regular language included in $\mathrm{Lan}(\mathcal{A})$ and obtained from small and accepting extended paths, by using Lemma 1.27. When $Q_0 \cap F \neq \emptyset$, we include a disjunct stating that all the values are equal to zero.

Exercises

Exercise 1.1. Show Lemma 1.2.

Exercise 1.2. Show that every formula in Presburger arithmetic is equivalent to a formula in prenex normal form, by using an algorithm that runs in polynomial time in the size of the input formula.

Exercise 1.3. Let φ be a Presburger formula with more than one free variable. Define a Presburger formula ψ such that ψ is satisfiable iff $[\![\varphi]\!]$ is finite.

Exercise 1.4. Let $\mathrm{AT}(\mathsf{x})$ be the set of atomic formulae of the form

$$a\mathsf{x} + b \leq a'\mathsf{x} + b'$$

where a, a', b and b' are natural numbers.

1. Show that every atomic formula $a\mathsf{x} + b \leq a'\mathsf{x} + b'$ is equivalent to an atomic formula of the form either $\top$, $\bot$, $\mathsf{x} \leq k$, or $\mathsf{x} \geq k$ with $k \in \mathbb{N}$.
2. Show that for every formula ψ that is a Boolean combination of atomic formulae of the form either $\top$, $\bot$ or $\mathsf{x} \leq k$, $[\![\varphi]\!]$ is a set of natural numbers equal to a finite union of intervals $\bigcup_i I_i$ such that each I_i is of the form either $[k_1, k_2]$ or $[k_1, +\infty[$ with $k_1, k_2 \in \mathbb{N}$.
3. Conclude that $\exists\, \mathsf{z}\ \mathsf{x} = 2\mathsf{z}$ cannot be equivalent to a Boolean combination of atomic formulae of the form $a\mathsf{x} + b \leq a'\mathsf{x} + b'$

Exercise 1.5. Let us consider the following fragment of $\mathrm{FO}(\mathbb{N})$:

$$\varphi ::= \ \top \ \mid \bot \mid \ \mathsf{x} \equiv_k \mathsf{y} \mid \ \mathsf{x} \equiv_k c \mid \ \mathsf{x} \leq c \mid \ \mathsf{x} = \mathsf{y} \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \exists \mathsf{x}\, \varphi$$

where x, y are variables, $k \geq 2$ and $c \geq 0$.

1. Show that every formula in that fragment is equivalent to a Boolean combination of atomic formulae of one of the forms below: $\mathsf{x} \equiv_k c$, $\mathsf{x} \leq c$ and $\mathsf{x} = \mathsf{y}$.

2. Show that the satisfiability problem is PSPACE-hard.
3. What about PSPACE-easiness?

Exercise 1.6. Show that ultimately periodic sets are closed under union, intersection and complementation.

Exercise 1.7. Compute a quantifier-free formula equivalent to the formula below following the procedure to eliminate quantifiers.

$$\exists\, \mathsf{z}_1, \mathsf{z}_2, \mathsf{z}_3\ (\mathsf{x}_1 = 3 + \mathsf{z}_1 - \mathsf{z}_2) \wedge (\mathsf{x}_2 = 3 + \mathsf{z}_2 + \mathsf{z}_3) \wedge\ (2 + \mathsf{z}_1 - \mathsf{z}_2 \geq 0).$$

Exercise 1.8. Let $X \subseteq \mathbb{N}^2$ be a semilinear set. Show that $\{k \in \mathbb{N} : \langle k, k' \rangle \in X\}$ (projection) is a semilinear subset of $\mathbb{N}$.

Exercise 1.9. Define a Presburger formula φ such that $[\![\varphi]\!] = \{\langle n_1, n_2 \rangle \in \mathbb{N} \times \mathbb{N} : n_1 \times n_2 \text{ is odd}\}$.

Exercise 1.10. Show that any arithmetic progression (viewed as a set of natural numbers) can be defined in Presburger arithmetic.

Exercise 1.11. Show that semilinear sets are Presburger definable.

Exercise 1.12. Let $X, Y \subseteq \mathbb{N}^n$. We define $X + Y$ as the set $\{\mathbf{x} + \mathbf{y} : \mathbf{x} \in X,\ \mathbf{y} \in Y\}$. Show that if X and Y are semilinear, then $X + Y$ is also semilinear.

Exercise 1.13. Let $X \subseteq \mathbb{N}$ be a set of natural numbers such that there are $a, b \in \mathbb{N}$ and $b > 0$ such that for every $n \in X$ such that $n > a$, we have $n + b \in X$. Show that X is semilinear.

Exercise 1.14. As stated in the proof of Theorem 1.23, show that every atomic formula from Presburger arithmetic defines a semilinear set.

Exercise 1.15. FO($\mathbb{Z}$) is defined as FO($\mathbb{N}$) except that variable valuations have domain $\mathbb{Z}$ instead of $\mathbb{N}$. In particular, terms, atomic formulae and formulae are identical.

The goal of this exercise is to show decidability of FO($\mathbb{Z}$) by translating the satisfiability problem for FO($\mathbb{Z}$) to the satisfiability problem for FO($\mathbb{N}$). A negative integer n is encoded by $-2n + 1$ whereas a non-negative integer m is encoded by $2m$.

1. Show in FO($\mathbb{Z}$) that every atomic formula $t \leq t'$ has an equivalent formula that uses only atomic formulae (or abbreviations) of the form either (1) $\mathsf{x} \geq 0$ or (2) $t = t'$.
2. Let us define a map $\mathfrak{g}$ from FO($\mathbb{Z}$) restricted to atomic formulae of the form (1) or (2) that is homomorphic for Boolean connectives and quantifiers such that $\mathsf{x} \geq 0$ is translated into $\mathsf{x} \equiv_2 0$. It remains to define the translation for atomic formulae of the form (2).

An atomic formula of the form

$$\sum_{j\in[1,n]} a_j \mathsf{x}_j = b$$

with $a_j \in \mathbb{Z}$ and $b \in \mathbb{Z}$ is encoded by the following disjunction:

$$\bigvee_{\mathbf{p}\in\{0,1\}^n} \exists\, \mathsf{y}_1, \ldots, \mathsf{y}_n\, (\bigwedge_i \psi(i, \mathbf{p}(i))) \wedge \sum_{j\in[1,n]} \varepsilon(\mathbf{p}(j), a_j)\mathsf{y}_j = b$$

where

- $\varepsilon(1, a)$ is equal to a and $\varepsilon(0, a)$ is equal to $-a$.
- $\psi(j, 0) =$ '$\mathsf{x}_j = 2\mathsf{y}_j + 1$' and $\psi(j, 1) =$ '$\mathsf{x}_j = 2\mathsf{y}_j$'.

Evaluate the size of $\mathfrak{g}(\varphi)$ with respect to the size of φ.

3. Given a formula $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ and its translation $\psi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$, show that

$$[\![\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)]\!] = \{\mathfrak{f}(\mathbf{x}) \in \mathbb{Z}^n : \mathbf{x} \in [\![\psi(\mathsf{x}_1, \ldots, \mathsf{x}_n)]\!]\}$$

where $\mathfrak{f}(\mathbf{x})(i) = \frac{\mathbf{x}(i)}{2}$ if $\mathbf{x}(i)$ is even, otherwise $\mathfrak{f}(\mathbf{x})(i) = -\frac{\mathbf{x}(i)-1}{2}$.

4. Conclude that the satisfiability problem for FO($\mathbb{Z}$) is decidable.

Bibliographical Notes

Presburger arithmetic (FO($\mathbb{N}$)) has been introduced by Mojzesz Presburger in (Presburger, 1929) where it is shown decidable by quantifier elimination. This decidability result on the theory of addition is regarded today as a major result in mathematics and computer science.

Quantifier elimination The approach presented in Section 1.3 is inspired from Cooper's elimination procedure (Cooper, 1972) in which when considering $\exists$ x ψ with quantifier-free ψ, we do not assume that ψ is in disjunctive normal form (a disjunction of conjuncts, with conjuncts made of literals). This is a remarkable difference with the original algorithm presented in (Presburger, 1929), apart from the fact that it was also required that the quantified formula has to be in prenex normal form. Indeed, transforming a propositional formula into an equivalent formula in disjunctive normal form may cause an exponential blow-up. As noted in (Stansifer, 1984) that contains a translation in English of Presburger's paper (Presburger, 1929), Presburger's procedure is very close to an algorithm that can be implemented even though Cooper's algorithm is much more efficient. A more advanced improvement of the procedure can be found in (Reddy and Loveland, 1978) whereas recent developments propose a lazy approach for quantifier elimination (Monniaux, 2010) (by contrast to the standard eager approach that consists in eliminating quantifiers by blocks).

Complexity of Presburger Arithmetic Decidability of Presburger arithmetic stated in Theorem 1.11 has been shown in (Presburger, 1929). The satisfiability problem for Presburger arithmetic can be solved in triple exponential time (Oppen, 1978) by analyzing the quantifier elimination procedure described in (Cooper, 1972). Besides, the satisfiability problem for Presburger arithmetic is shown 2ExpTime-hard in (Fischer and Rabin, 1974) and in 2ExpSpace in (Ferrante and Rackoff, 1979, Chapter 3, Theorem 1.7). An exact complexity charaterisation is provided in (Berman, 1980) (double exponential time on alternating Turing machines with linear amounts of alternations). Due to the wide range of applications for Presburger arithmetic, computational complexity of numerous fragments has been also characterised, see e.g. (Grädel, 1988). Moreover, its restriction to quantifier-free formulae is NP-complete (Papadimitriou, 1981) (see also (Borosh and Treybig, 1976)). Theorem 1.9 is a consequence of developments in (Borosh and Treybig, 1976; Papadimitriou, 1981; Schrijver, 1986). These works propose different ways to measure the size of small solutions; each measure makes sense depending on the relevant parameters. For instance, the work on one-counter automata in (Göller et al., 2012) takes advantage of bounds established in (Schrijver, 1986). Note also that Theorem 1.12 is due to (Grädel, 1988, Theorem 3.7). Let us conclude this paragraph by a nice result due to (Lenstra, 1983; Scarpellini, 1984): given a fixed $k \geq 1$, the satisfiability problem for the quantifier-free fragment restricted to formulae with at most k variables and with atomic formulae of the form $t \leq t'$ can be solved in polynomial time.

Ubiquity of Presburger Arithmetic Arithmetical constraints expressed in Presburger arithmetic are used in numerous formalisms including counter machines, modal and temporal logics (Bouajjani et al., 1995; Comon and Cortier, 2000; Demri and Gascon, 2008; Demri and Lugiez, 2010), languages to specify XML documents (Zilio and Lugiez, 2003; Seidl et al., 2007). This list is certainly not exhaustive.

Techniques for the verification of infinite-state systems also used intensively this formalism either to get decision procedures (Fribourg and Olsén, 1997; Leroux, 2003; Schuele, 2007; Leroux, 2010) or to represent symbolically infinite sets of configurations (Comon and Jurski, 1998).

Semilinear sets Theorem 1.22 and Theorem 1.23 are due to (Ginsburg and Spanier, 1966) (see also (Reutenauer, 1990)). An alternative proof can be found in (Kracht, 2002).

Parikh's Theorem Parikh's remarkable result states that the Parikh image of any context-free language is semilinear (Parikh, 1966) and that its representation is computable with a pushdown automaton. By the way, Parikh's proof can be also found in (Kozen, 1997, Chapter H). An alternative proof is also given in (Esparza, 1997) based on the result that the reachability relation for communication-free Petri nets is computable and semilinear.

For regular languages, it is possible to impose constraints on its Parikh image about the size of the representation in terms of basis and periods. Let $\mathcal{A}$ be a finite-state automaton with a set of states Q and a finite alphabet Σ. The Parikh image of $\mathrm{L}(\mathcal{A})$, a subset of $\mathbb{N}^{\mathrm{card}(\Sigma)}$, is a finite union $X_1 \cup \cdots \cup X_m$ of linear sets of the form $X_i = \{\mathbf{b} + \sum_{j=1}^{h} \lambda_j \mathbf{p}_j : \lambda_j \geq 0\}$ where $\mathbf{b}$ and the $\mathbf{p}_j$'s are in $\{0, \ldots, \mathrm{card}(Q)\}^{\mathrm{card}(\Sigma)}$ by (Seidl et al., 2004, Theorem 1). Consequently, h is bounded by $(\mathrm{card}(Q)+1)^{\mathrm{card}(\Sigma)}$. It is also possible to build $\varphi_{\mathcal{A}}$ in polynomial-time in the size of $\mathcal{A}$ such that $\Pi(\mathrm{Lan}(\mathcal{A})) = [\![\varphi_{\mathcal{A}}]\!]$ (see e.g. (Seidl et al., 2004)).

Theorem 1.28 is shown in (Blattner and Latteux, 1981, Theorem 2) in its more general form for context-free languages, see also (Esparza et al., 2011).

Tools Even though Cooper's elimination procedure is much more efficient than Presburger's algorithm, as far as we can judge, the first implemented algorithm for deciding Presburger arithmetic is due to Martin Davis who wrote a program based on Presburger's algorithm in 1954. As written in (Davis, 1999) from the original Davis' paper, "its great triumph was to prove that the sum of two even numbers is even". Since then, deciding fragments of Presburger arithmetic is an essential part of many reasoning tasks, for instance to verify properties on programs, see e.g. (Shostak, 1979) and (Suzuki and Jefferson, 1980) for an early use of Presburger arithmetic for formal verification, see also (Fribourg, 2000).

Most well-known Satisfiability Modulo Theories (SMT) solvers can deal with quantifier-free formulae, also known as linear arithmetic (LIA). For instance, this includes Z3 (de Moura and Björner, 2008), CVC4 (Barrett et al., 2011) and Alt-Ergo (Conchon, 2012), to cite a few of them; see also Pugh's Omega test to deal with quantifier-free formulae (Pugh, 1992). Another approach using binary decision diagrams (BDD) in automata is proposed in (Ganesh et al., 2002) in order to solve the satisfiability problem for quantifier-free formulae. However, dealing with quantifiers is usually a difficult task for SMT solvers that are better tailored to theory reasoning. SMT solver Z3 is one of the rare SMT solvers that can handle quantified Presburger formulae.

Automata-based decision procedures The satisfiability problem for $\mathrm{FO}(\mathbb{N})$ is decidable by quantifier elimination. An alternative approach consists in reducing the satisfiability problem to the nonemptiness problem for finite-state automata. The use of automata for logical decision problems goes back to (Büchi, 1960a) and in the seminal paper (Büchi, 1960b), it is already mentioned how to use the automata-based approach to deal with Presburger arithmetic.

We have seen that a formula $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_d)$ with $d \geq 1$ defines a subset of $\mathbb{N}^d$, namely $[\![\varphi]\!] \subseteq \mathbb{N}^d$, that corresponds to the set of variable valuations that make φ true. The automata-based approach consists in representing the tuples in $[\![\varphi]\!]$ by a regular language that can be effectively defined, for instance with the help of a finite-state automaton. A correspondence is actually used between logical connectives (conjunction, negation, quantification, ...) and operations on automata (intersection, complementation, projection, ...), see e.g. (Boudet and Comon, 1996).

Extensions First-order theories of arithmetic have been intensively studied in the past, and Presburger arithmetic is one of them. Below, we list decidability/undecidability results about extensions of $\mathrm{FO}(\mathbb{N})$.

- Undecidability of the first-order theory of $\langle \mathbb{N}, \leq, \times, + \rangle$ ($\mathrm{FO}(\mathbb{N})$ with multiplication) can be found in (Tarski, 1953). Actually, this result is a consequence of Gödel first incompleteness theorem (Gödel, 1931), see e.g. (Church, 1936). However, the monograph (Tarski, 1953) (and more specifically Chapter II) presents a proof technique to show undecidability of first-order theories of arithmetic. Apart from the proof technique itself, it is also of great interest for its rich amount of bibliographical references (Tarski, 1953).

- FO($\mathbb{N}$) extended with divisibility predicate (usually written $|$) is undecidable, see e.g. (Robinson, 1949) whereas its existential restriction is decidable (Lipshitz, 1978). Existential fragment corresponds to formulae in prenex normal form in which the quantifier prefix contains only the existential quantifier.
- FO($\mathbb{N}$) extended with the unary function symbol V_k so that $V_k(n)$ is the greatest power of k that divides n. This extension is decidable by (Büchi, 1960b) by using Büchi's encoding of FO($\mathbb{N}$) in weak monadic second-order logic. Each natural number is encoded by a finite set of natural numbers that corresponds to non-zero positions in its binary representation. Number 11 is encoded by $\{0, 1, 3\}$ for instance.
- FO($\mathbb{N}$) extended with relative primeness predicate (usually written $\perp$) is undecidable (Woods, 1981).
- FO($\mathbb{N}$) extended with the unary prime predicate is undecidable (Woods, 1981).
- FO($\mathbb{N}$) with unary predicates is undecidable (Halpern, 1991). Actually, it is shown in (Halpern, 1991), that adding to FO($\mathbb{N}$) the possibility to quantify over a single monadic second-order variable P leads to undecidability. In such an extension, formulae of the form $\forall\, \mathsf{P}\ (\forall \mathsf{x}\ \varphi(\mathsf{x}) \Rightarrow (\mathsf{x} \in \mathsf{P}))$ are allowed, where for instance $\varphi(\mathsf{x})$ is a formula from FO($\mathbb{N}$). Note the new quantification $\forall\, \mathsf{P}$ and the new atomic formula $\mathsf{x} \in \mathsf{P}$.

A survey about decidability of first-order fragments of arithmetic, including extensions of Presburger arithmetic, can be found in (Bès, 2002) as well as plenty of bibliographical references on the subject.

REFERENCES

Barrett, C., Conway, C., Deters, M., Hadarean, L., Jovanovic, D., King, T., Reynolds, A., and Tinelli, C., 2011. CVC4. In *CAV 2011*, volume 8606 of *Lecture Notes in Computer Science*, pages 171–177. Springer. doi:10.1007/978-3-642-22110-1_14. Cited on page 39.

Berman, L., 1980. The complexity of logical theories. *Theoretical Computer Science*, 11(1):71–78. doi:10.1016/0304-3975(80)90037-7. Cited on page 38.

Bès, A., 2002. A survey of arithmetical definability. In *A tribute to Maurice Boffa*, pages 1–54. Soc. Math. Belgique. Cited on page 40.

Blattner, M. and Latteux, M., 1981. Parikh-Bounded Languages. In *ICALP '81*, volume 115 of *Lecture Notes in Computer Science*, pages 316–323. Springer. doi:10.1007/3-540-10843-2_26. Cited on page 39.

Boigelot, B. and Wolper, P., 2002. Representing arithmetic constraints with finite automata: an overview. In Stuckey, P., editor, *ICLP 2002*, volume 2401 of *Lecture Notes in Computer Science*, pages 1–19. Springer. doi:10.1007/3-540-45619-8_1. Cited on pages 24, 25.

Borosh, I. and Treybig, L.B., 1976. Bounds on positive integral solutions of linear diophantine equations. *Proc. Amer. Math. Soc.*, 55(2):299–304. doi:10.1090/S0002-9939-1976-0396605-3. Cited on page 38.

Bouajjani, A., Echahed, R., and Habermehl, P., 1995. On the verification problem of nonregular properties for nonregular processes. In *LICS'95*, pages 123–133. Cited on page 38.

Boudet, A. and Comon, H., 1996. Diophantine equations, Presburger arithmetic and finite automata. In *CAAP '96*, volume 1059 of *Lecture Notes in Computer Science*, pages 30–43. Springer. doi:10.1007/3-540-61064-2_27. Cited on pages 21, 23, 24, 25, 39.

Büchi, J.R., 1960a. On a decision method in restricted second-order arithmetic. In *Logic, Methodology, and Philosophy of Science*, pages 1–11. doi:10.1007/978-1-4613-8928-6_23. Cited on pages 21, 22, 39.

Büchi, J.R., 1960b. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:66–92. doi:10.1002/malq.19600060105. Cited on pages 21, 39, 40.

Church, A., 1936. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363. doi:10.2307/2371045. Cited on page 39.

Comon, H. and Jurski, Y., 1998. Multiple counters automata, safety analysis, and Presburger arithmetic. In *CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer. doi:10.1007/BFb0028751. Cited on page 38.

Comon, H. and Cortier, V., 2000. Flatness is not a weakness. In *CSL'00*, volume 1862 of *Lecture Notes in Computer Science*, pages 262–276. Springer. Cited on page 38.

Conchon, S., 2012. SMT Techniques and their Applications: from Alt-Ergo to Cubicle. *Habilitation à Diriger des Recherches, Université Paris-Sud*. Cited on page 39.

Cooper, D.C., 1972. Theorem proving in arithmetic without multiplication. In Meltzer, B. and Michie, D., editors, *Machine Intelligence*, volume 7, chapter 5, pages 91–99. Edinburgh University Press. Cited on pages 21, 37, 38.

Davis, M., 1999. The early history of automated deduction. In *Handbook of Automated Reasoning*,

pages 3–15. Elsevier. doi:10.1016/B978-044450813-3/50003-5. Cited on page 39.

de Moura, L. and Björner, N., 2008. Z3: An Efficient SMT Solver. In *TACAS'08*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer. Cited on page 39.

Demri, S. and Gascon, R., 2008. Verification of qualitative Z constraints. *Theoretical Computer Science*, 409(1):24–40. Cited on page 38.

Demri, S. and Lugiez, D., 2010. Complexity of modal logics with Presburger constraints. *Journal of Applied Logic*, 8(3):233–252. doi:10.1016/j.jal.2010.03.001. Cited on page 38.

Demri, S. and Poitrenaud, D., 2011. Verification of infinite-state systems. In Haddad, S., Kordon, F., Pautet, L., and Petrucci, L., editors, *Models and Analysis for Distributed Systems*, chapter 8, pages 221–269. Wiley. Cited on page 21.

Durand-Gasselin, A. and Habermehl, P., 2010. On the use of non-deterministic automata for Presburger arithmetic. In *CONCUR 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 373–387. Springer. doi:10.1007/978-3-642-15375-4_26. Cited on page 25.

Esparza, J., 1997. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundamenta Informaticae*, 31(1):13–25. doi:10.3233/FI-1997-3112. Cited on page 38.

Esparza, J., Ganty, P., Kiefer, S., and Luttenberger, M., 2011. Parikh's Theorem: A simple and direct automaton construction. *Information Processing Letters*, 111:614–619. doi:10.1016/j.ipl.2011.03.019. Cited on page 39.

Ferrante, J. and Rackoff, C., 1979. *The Computational Complexity of Logical Theories*, volume 718 of *Lecture Notes in Mathematics*. Springer. Cited on page 38.

Fischer, M.J. and Rabin, M.O., 1974. Super-exponential complexity of Presburger arithmetic. In *Complexity of Computation*, volume 7 of *SIAM-AMS proceedings*, pages 27–42. American Mathematical Society. http://www.lcs.mit.edu/publications/pubs/ps/MIT-LCS-TM-043.ps. Cited on page 38.

Fribourg, L., 2000. Petri nets, flat languages and linear arithmetic. In *9th International Workshop on Functional and Logic Programming, WFLP'2000*, pages 344–365. Cited on page 39.

Fribourg, L. and Olsén, H., 1997. Proving safety properties of infinite state systems by compilation into Presburger arithmetic. In *CONCUR '97*, volume 1243 of *Lecture Notes in Computer Science*, pages 213–227. Springer. doi:10.1007/3-540-63141-0_15. Cited on page 38.

Ganesh, V., Berezin, S., and Dill, D., 2002. Deciding Presburger arithmetic by model checking and comparisons with other methods. In *FMCAD 2002*, volume 2517 of *Lecture Notes in Computer Science*, pages 171–186. Springer. doi:10.1007/3-540-36126-X_11. Cited on page 39.

Ginsburg, S. and Spanier, E.H., 1966. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296. http://projecteuclid.org/euclid.pjm/1102994974. Cited on page 38.

Gödel, K., 1931. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. *Monatshefte für Mathematik und Physik*, 38:173–198. Cited on page 39.

Göller, S., Haase, C., Ouaknine, J., and Worrell, J., 2012. Branching-time model checking of parametric one-counter automata. In *FOSSACS 2012*, volume 7213 of *Lecture Notes in Computer Science*, pages 406–420. Springer. doi:10.1007/978-3-642-28729-9_27. Cited on page 38.

Grädel, E., 1988. Subclasses of Presburger arithmetic and the polynomial-time hierarchy. *Theoretical Computer Science*, 56(3):289–301. doi:10.1016/0304-3975(88)90136-3. Cited on page 38.

Halpern, J.Y., 1991. Presburger arithmetic with unary predicates is Π^1_1-complete. *Journal of Symbolic Logic*, 56(2):637–642. doi:10.2307/2274706. Cited on page 40.

Klaedtke, F., 2004a. On the automata size for Presburger arithmetic. In *LICS 2004*, pages 110–119. IEEE. doi:10.1109/LICS.2004.1319605. Cited on page 22.

Klaedtke, F., 2004b. *Automata-based decision procedures for weak arithmetics*. PhD thesis, Institut für Informatik, Albert-Ludwigs-Universität, Freiburg. Cited on pages 23, 25.

Kozen, D., 1997. *Automata and Computability*. Springer. Cited on page 38.

Kracht, M., 2002. A new proof of a theorem by Ginsburg and Spanier. Manuscript, Dept. Linguistics, UCLA. Cited on page 38.

Legay, A., 2008. *Generic methods for the verification of infinite-state systems.* PhD thesis, Université de Liège. Cited on page 23.

Lenstra, H., 1983. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548. doi:10.1287/moor.8.4.538. Cited on page 38.

Leroux, J., 2003. *Algorithmique de la vérification des systèmes à compteurs. Approximation et accélération. Implémentation de l'outil FAST.* PhD thesis, ENS de Cachan, France. Cited on pages 22, 25, 38.

Leroux, J., 2010. The general vector addition system reachability problem by Presburger inductive invariants. *Logical Methods in Computer Science*, 6(3). doi:10.2168/LMCS-6(3:22)2010. Cited on page 38.

Lipshitz, L., 1978. The Diophantine problem for addition and divisibility. *Transactions of the American Mathematical Society*, pages 271–283. doi:10.2307/1998219. Cited on page 40.

Monniaux, D., 2010. Quantifier elimination by lazy model enumeration. In *CAV 2010*, volume 6174 of *Lecture Notes in Computer Science*, pages 585–599. Springer. doi:10.1007/978-3-642-14295-6_51. Cited on page 37.

Oppen, D.C., 1978. A $2^{2^{2pn}}$ upper bound on the complexity of Presburger arithmetic. *Journal of Computer and System Sciences*, 16(3):323–332. doi:10.1016/0022-0000(78)90021-1. Cited on page 38.

Papadimitriou, C., 1981. On the complexity of integer programming. *Journal of the ACM*, 28(4): 765–768. Cited on page 38.

Parikh, R., 1966. On context-free languages. *Journal of the ACM*, 13(4):570–581. doi:10.1145/321356.321361. Cited on page 38.

Presburger, M., 1929. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus du premier congrès de mathématiciens des Pays Slaves, Warszawa*, pages 92–101. Cited on pages 1, 37, 38.

Pugh, W., 1992. A practical algorithm for exact array dependence analysis. *Communications of the ACM*, 35(8):102–114. doi:10.1145/135226.135233. Cited on page 39.

Reddy, C. and Loveland, W., 1978. Presburger arithmetic with bounded quantifier alternation. In *STOC'78*, pages 320–325. ACM press. Cited on pages 21, 37.

Reutenauer, C., 1990. *The mathematics of Petri nets.* Masson and Prentice. Cited on page 38.

Robinson, J., 1949. Definability and decision problems in arithmetic. *Journal of Symbolic Logic*, 14 (2):98–114. doi:10.2307/2266510. Cited on page 40.

Scarpellini, B., 1984. Complexity of subcases of Presburger arithmetic. *Transactions of the American Mathematical Society*, 284:203–218. doi:10.2307/1999283. Cited on page 38.

Schrijver, A., 1986. *Theory of Linear and Integer Programming.* John Wiley & Sons. Cited on page 38.

Schuele, T., 2007. *Verification of Infinite State Systems Using Presburger Arithmetic.* PhD thesis, Department of Computer Science, University of Kaiserslautern, Germany, Kaiserslautern, Germany. Cited on page 38.

Schuele, T. and Schneider, K., 2007. Verification of data paths using unbounded integers: automata strike back. In *HVC 2006*, volume 4383 of *Lecture Notes in Computer Science*, pages 65–80. Springer. doi:10.1007/978-3-540-70889-6_5. Cited on page 25.

Seidl, H., Schwentick, T., Muscholl, A., and Habermehl, P., 2004. Counting in trees for free. In *ICALP'04*, volume 3142 of *Lecture Notes in Computer Science*, pages 1136–1149. Springer. Cited on page 38.

Seidl, H., Schwentick, T., and Muscholl, A., 2007. Counting in trees. In *Logic and Automata. History and Perspectives*, volume 2 of *Texts in Logic and Games*, chapter 18, pages 575–612. Amsterdam University Press. Cited on page 38.

Shostak, R., 1979. A practical decision procedure for arithmetic with function symbols. *Journal of the ACM*, 26(2):351–360. doi:10.1145/322123.322137. Cited on page 39.

Stansifer, R., 1984. Presburger's article on integer arithmetic: remarks and translation. Technical

Report TR-84-639, Cornell University, Department od Computer Science. Cited on page 37.

Suzuki, N. and Jefferson, D., 1980. Verification decidability of Presburger array programs. *Journal of the ACM*, 27(1):191–205. doi:10.1145/322169.322185. Cited on page 39.

Tarski, A., 1953. *Undecidable Theories*. Studies in Logic and the Foundations of Mathematics. North-Holland. In collaboration with A. Mostowski. and R. Robinson. Cited on page 39.

Wolper, P. and Boigelot, B., 1995. An automata-theoretic approach to Presburger arithmetic constraints. In *SAS '95*, volume 983 of *Lecture Notes in Computer Science*, pages 21–32. Springer. Cited on page 21.

Woods, A., 1981. *Some problems in logic and number theory, and their connections*. PhD thesis, University of Manchester. Cited on page 40.

Zilio, S.D. and Lugiez, D., 2003. XML schema, tree logic and sheaves automata. In *RTA'03*, volume 2706 of *Lecture Notes in Computer Science*, pages 246–263. Springer. Cited on page 38.