



ON THEORIES MIXING ARITHMETIC AND
UNINTERPRETED PREDICATES

by

Baptiste VERGAIN

*A thesis submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy (PhD) in Engineering and Technology Sciences*

SEPTEMBER 2026

supervised by

Prof. Bernard BOIGELOT *and by*
Prof. Pascal FONTAINE



DOCTORAL COLLEGE IN ELECTRICAL ENGINEERING AND
COMPUTER SCIENCE

Résumé

Cette thèse traite de la décidabilité de théories combinant contraintes arithmétiques et symboles non-interprétés, ainsi que de la conception de procédures de décision dédiées. La présence de contraintes arithmétiques ou de symboles non-interprétés permet souvent d'obtenir des théories très expressives, mais la combinaison des deux mène rapidement à l'indécidabilité. Une bonne illustration de cet équilibre fragile se trouve dans le fait qu'il suffit d'étendre l'arithmétique de Presburger avec un seul prédicat unaire non-interprété pour la rendre indécidable.

Les premières contributions de ce travail consistent à établir un seuil minimal pour la décidabilité, et un seuil maximal pour l'indécidabilité de théories faisant intervenir ensemble arithmétique et symboles non-interprétés, affinant ainsi la frontière de la décidabilité. Dans un premier temps, nous démontrons que l'extension de la logique de différence interprétée sur les réels avec l'ajout d'un unique prédicat unaire non-interprété est indécidable. Nous établissons ensuite la décidabilité de la théorie monadique du premier ordre sur les ordres linéaires $\text{MFO}(<)$, interprétée sur les nombres réels équipés de leur relation d'ordre habituelle, et étendue avec des contraintes de différence limitées aux variables entières. Plus précisément, nous montrons comment traduire toute formule de cette théorie en une formule équisatisfiable décrite dans $\text{MFO}(<)$ interprétée sur les réels. Nous développons également une preuve alternative et autonome de la décidabilité de $\text{MFO}(<)$ interprétée sur $\mathbb{R}$. Cette preuve met en exergue une classe de modèles suffisante vis-à-vis de la satisfiabilité, qui pourrait s'avérer utile dans la conception d'une procédure de décision efficace pour cette théorie.

Par la suite, nous explorons la conception concrète d'une procédure de décision par automates pour $\text{MFO}(<)$ interprétée sur différentes classes d'ordres linéaires. Notre approche s'inspire de la procédure de décision décrite par Büchi pour décider S1S qui utilise les automates sur mots infinis éponymes. Une procédure de décision par automate procède en construisant inductivement un automate reconnaissant les modèles d'une formule donnée en entrée, en suivant sa structure syntaxique. Plus précisément, on définit un automate pour chaque atome de la théorie, et on décrit des opérations sur automates correspondant aux différents opérateurs et connecteurs logiques. La dernière étape consiste à tester si le langage accepté par l'automate résultant est vide ou non. Afin de couvrir des domaines denses tels que $\mathbb{R}$ et $\mathbb{Q}$, ou bien des classes d'ordres linéaires

telle que la classe de tous les ordinaux, nous nous appuyons sur les automates sur ordres linéaires introduits par Bruyère et Carton, qui généralisent les notions d'automates sur mots finis, infinis, et transfinis. Ces automates lisent des mots indexés par des ordres linéaires quelconques, c'est-à-dire, des mappings des éléments d'un tel ordre vers un alphabet fini. Une des difficultés qui survient lorsque l'on utilise ces automates sur ordres linéaires pour concevoir une procédure de décision, est que l'on ne peut pas s'appuyer sur la complémentation pour traiter la négation et la quantification universelle, comme c'est habituellement le cas, ces automates n'étant pas fermés par complémentation sur la classe de tous les ordres linéaires.

Afin de surmonter cet obstacle, nous décrivons la structure d'une procédure de décision affranchie de toute complémentation d'automate. Nous présentons ensuite un algorithme polynomial pour tester si un automate accepte au moins un mot indexé par l'ordre des réels, ou l'ordre des rationnels. Enfin, nous développons un algorithme pour calculer directement le résultat d'une quantification universelle du premier ordre sur un automate de Büchi, avant de discuter de la généralisation de cette opération sur les automates sur ordres linéaires.

Abstract

This thesis addresses the decidability of theories combining arithmetic and uninterpreted symbols, and the design of effective decision procedures for them. The presence of arithmetic or uninterpreted symbols often yields theories with a strong expressive power, and the combination of these elements quickly leads to undecidability. The undecidability of Presburger arithmetic extended with a single unary uninterpreted predicate serves as a good illustration of this delicate balance.

The first contributions of this work consist in establishing a lower bound for decidability, and an upper bound for undecidability of theories mixing arithmetic and uninterpreted symbols, therefore refining the frontier of decidability. On the one hand, we show that extending difference logic over the reals with a single unary uninterpreted predicate makes it undecidable. On the other hand, we prove that the monadic first-order theory of order over the reals, with difference constraints restricted to integer variables only is decidable. More precisely, we develop an algorithm to translate any formula from the latter theory into an equisatisfiable formula in the monadic first-order theory of order $\text{MFO}(<)$ interpreted over the reals. We also give an alternate self-contained proof of the decidability of $\text{MFO}(<)$ over the reals, providing insights on a sufficient class of models w.r.t. satisfiability, which may prove helpful in the design of an effective decision procedure for this theory.

We then investigate how to design an automata-based decision procedure for $\text{MFO}(<)$, interpreted over various classes of domains. We take inspiration from the procedure introduced by Büchi to decide S1S that relies on the eponymous infinite-word automata. The principle of such an automata-based decision procedure is to inductively build an automaton recognizing the models of a given input formula, following its syntactic structure. More precisely, one defines an automaton for each atom of the theory, and provides operations on automata matching Boolean connectives and other logical operators. The last step consists in checking whether the resulting automaton accepts a non-empty language. In order to deal with dense domains such as $\mathbb{R}$ or $\mathbb{Q}$, or classes of domains such as the class of all ordinals, we use automata on linear orderings, introduced by Bruyère and Carton, that generalize finite-word, infinite-word, and ordinal automata. These automata read words indexed by arbitrary linear orderings, that is, mappings from the elements of such orderings to a finite alphabet. One difficulty that arises when following this approach with automata on linear ordering

is that we cannot rely on complementation, as is usually the case, to deal with negation and universal quantification. Indeed, it is known that these automata are not closed under complementation over the class of all linear orderings.

To overcome this obstacle, we lay out the overall structure of a decision procedure free from automata complementation. We then describe a polynomial algorithm to decide the non-emptiness of the language accepted by an automaton on linear orderings restricted to the orderings of the reals and rationals. Finally, we provide an algorithm to directly compute the effect of first-order universal quantification on Büchi automata, and then discuss the generalization of this operation to automata on linear orderings.

Acknowledgments

Acknowledgments will come in the final version.

*This thesis reproduces in parts works partially supported by the FNRS-DFG
PDR Weave (SMT-ART) grant 40019202, and by an Amazon Research
Award, Fall 2022 CFP.*

Contents

Introduction	11
1 Preliminaries	15
1.1 Logic	15
1.1.1 Sets, Relations, and Functions	15
1.1.2 Logic Terminology	17
1.2 Theory of Order	22
1.2.1 Linear Orderings	22
1.2.2 Ordinal Numbers	24
1.2.3 Cardinal Numbers	24
1.2.4 Cofinality and Coinitiality	25
1.3 Languages, Regular Expressions, and Automata	25
1.3.1 Words Indexed by Linear Orderings and Regular Expressions	25
1.3.2 Automata	27
2 Drawing the Frontier of Decidability	35
3 Decidable and Undecidable Theories: Contributions	41
3.1 The Case of the Theory of Order over the Reals with Difference Logic over the Integers	41
3.1.1 Recognizing Integer Values	42
3.1.2 Translating Formulas	44
3.1.3 Establishing Equisatisfiability	44
3.2 The Case of Difference Logic over the Reals with One Predicate Symbol	45
3.2.1 Definitions	46
3.2.2 Encoding Runs	46
3.2.3 Defining the Support of the Encoding	47
3.2.4 Encoding a Configuration of the Turing Machine	51
3.2.5 Enforcing a Valid Run	52
3.3 An Alternate Proof of the Decidability of $\text{MFO}(\mathbb{R}, <)$	54
3.3.1 Additional Definitions	55
3.3.2 Elements of Model Theory	55

3.3.3	Constructible Chains	58
3.3.4	Deciding Satisfiability	73
4	Automata-Based Decision Procedures	83
4.1	Illustration of the Approach	83
4.2	Deciding MFO(<)	86
5	Non-Emptiness Test	95
5.1	State of the Art	96
5.2	Reachability over the Reals	96
5.2.1	Characterization	96
5.2.2	Algorithm	99
5.2.3	Correctness	102
5.2.4	Example	102
5.2.5	Complexity	103
5.2.6	Pseudo Code	104
5.3	Reachability over the Rationals	106
5.3.1	Algorithm	107
5.3.2	Example	108
5.4	Proof of Correctness	109
5.4.1	Soundness	109
5.4.2	Completeness	113
5.5	Additional Remarks	122
6	First-Order Universal Quantification	123
6.1	Principles	123
6.2	Quantification Over Natural Numbers	124
6.2.1	Principles	125
6.2.2	Construction	126
6.2.3	A Criterion for Accepting Runs	127
6.2.4	Computation Steps	129
6.2.5	Illustration	130
6.2.6	Proof of Correctness	130
6.2.7	Complexity	133
6.3	Towards an Extension for Other Linear Orderings	134
6.3.1	Reduction to General Automata Intersection	135
6.3.2	Hindrances	138
	Conclusion	141

Introduction

Context and Goals Effective decision procedures for arithmetic theories are essential to many domains, notably in verification, where it is common to rely on Satisfiability Modulo Theory (SMT) solvers [2, 3] to discharge proof obligations. This work is part of the long term goal to develop practical techniques to handle quantifiers within SMT solvers, in a context mixing arithmetic and uninterpreted predicates. We are convinced that a deep understanding of decision procedures for limited (but decidable) fragments will be helpful in the future to design such practical techniques. For instance, in a recent paper [61], automata-based decision procedures prove to be of help to improve quantified linear integer arithmetic reasoning in SMT solvers.

On its own, too much arithmetic expressive power leads to undecidability. For instance, Peano arithmetic — i.e., the first-order additive and multiplicative theory of the natural numbers — is undecidable [55, 68]. Undecidability arises even faster when arithmetic is combined with uninterpreted symbols, leaving little room of margin. As an example, extending Presburger arithmetic — i.e., the first-order additive theory of the natural numbers — with just one uninterpreted unary predicate makes it undecidable [42, 62, 66].

A famous and useful theory that mixes arithmetic and uninterpreted symbols while remaining decidable is the monadic second-order theory of $\mathbb{N}$ under one successor (S1S). The theory S1S is equivalent to the monadic second-order theory of order over the naturals $\text{MSO}(\mathbb{N}, <)$. The decidability of S1S has been established by Büchi [22] using the concept of infinite-word automaton. In order to decide whether a formula φ is satisfiable, the approach consists in building an automaton accepting the set of its models, encoded in a suitable way, and then checking that this automaton accepts a non-empty language. It is constructed by starting from elementary automata representing the atoms of φ , and then translating the effect of Boolean connectives and quantifiers into corresponding operations over automata.

Even though this approach has originally been introduced as a purely theoretical tool, it is applied in practice to obtain usable decision procedures for various logics. In particular, the tool MONA [69] uses this method to decide a restricted version of S1S, and tools such as LASH [16] and WALNUT [103] use a similar technique to decide extensions of Presburger arithmetic. The automata-based theorem prover PECAN [82], inspired by WALNUT, uses Büchi automata in order to deal with Sturmian words. The more recent tool KOFALA [1] provides

efficient implementation of complementation and inclusion checking of Büchi automata.

Theories related to the theory of order have been extensively studied, over various domains of interpretation. Büchi [23] showed that the monadic second-order theory of order over all ordinals strictly less than ω_2 is decidable. Over dense domains, Rabin [87] proved the decidability of the monadic second-order theory of order over $\mathbb{Q}$, but Shelah [105] showed on the other hand that the monadic second-order theory of order over $\mathbb{R}$ is undecidable. The monadic first-order — i.e., when second-order quantification is not allowed — theories of order over the reals [25] and over the class of all linear orderings [44, 73] are decidable.

Contributions The contributions of this work can be divided into two categories. First, we study the decidability of theories related to the monadic first-order theory $\text{MFO}(\mathbb{R}, <)$ of order interpreted over $\mathbb{R}$, mixing arithmetic and uninterpreted predicates. We show that extending $\text{MFO}(\mathbb{R}, <)$ with integer difference constraints — i.e., arithmetic constraints of the form $x - y \bowtie c$, where x and y are first-order integer variables, c is an integer constant, and $\bowtie$ ranges over $\{<, \leq, =, \geq, >\}$ — preserves decidability. More precisely, any formula in this extension can be translated into an equisatisfiable $\text{MFO}(\mathbb{R}, <)$ formula. However, we also show that extending $\text{MFO}(\mathbb{R}, <)$ with difference constraints over real variables makes it undecidable. We also provide an alternate self-contained proof of the decidability of $\text{MFO}(\mathbb{R}, <)$.

Contributions of the second category take the form of incremental steps towards the design of an effective automata-based decision procedure for the monadic theories of order interpreted over the reals, the rationals, or more general classes of domains. We lay out a precise structure for such a decision procedure, relying on automata on linear orderings introduced in [20] by Bruyère and Carton. They generalize the notions of finite-word, infinite-word and ordinal-word automata. Although initially restricted to scattered linear orderings (i.e., orderings that do not contain a dense subordering), these automata have later been extended to deal with all linear orderings [8]. Notably, a Kleene-like theorem asserting the equivalence between languages accepted by automata on linear orderings and languages described by an extended form of rational expressions is proved in [20, 8]. The strong connections between rational languages and the monadic second-order theory of linear orderings $\text{MSO}(<)$ is investigated in [5], namely, every language accepted by an automaton on linear orderings is definable in $\text{MSO}(<)$. The converse property however only holds when the orderings are countable and scattered. We then focus on two crucial steps of the procedure. First, we design non-emptiness tests specifically tailored for the domains $\mathbb{R}$ and $\mathbb{Q}$. We precisely describe effective polynomial algorithms. Second, we investigate how to conceive an operation on automata corresponding to a first-order universal quantification. We present an explicit algorithm when the domain is set to $\mathbb{N}$, and propose strategies for extending this result to larger classes of linear orderings.

Thesis Outline This thesis is structured as follows.

- In Chapter 1, we recall basic notions and define the notations used throughout this thesis. We start by introducing the fundamentals of logic focusing on first-order and second-order theories. We then turn to a presentation of the theory of order, introducing the notion of linear orderings and related concepts. Lastly, we define the notions of finite words, infinite words, and words indexed by arbitrary linear orderings. This is combined with a presentation of the corresponding notions of regular expressions, and the definitions of automata reading such words.
- In Chapter 2, we recall the state of the art regarding the decidability of logics, with a special focus on theories involving either arithmetic, uninterpreted monadic predicates, or a combination of both. Our goal being to draw a precise picture of the frontier of decidability for theories related to the monadic second-order theory of order.
- In Chapter 3, we present some of our contributions that further refine the frontier of decidability presented in the previous chapter. Namely, we start by proving the decidability of the monadic first-order theory of order over the reals, with difference constraints restricted to integer variables. We then prove the undecidability of the monadic first-order theory of order over the reals with unrestricted difference constraints. Finally, we provide an alternate self-contained proof of the decidability of the monadic first-order theory of order over the reals.
- In Chapter 4, we introduce the notion of an automata-based procedure to decide a theory. We recall the procedure introduced by Büchi to decide S1S, and then give the outline of an automata-based decision procedure for the monadic first-order theory of order interpreted over various classes of domains.
- Chapter 5 is dedicated to the design of algorithms to decide if a given automaton accepts at least one word indexed by the reals or the rational.
- Chapter 6 deals with the problem of computing the effect of a first-order universal quantification on an automaton recognizing the set of models of a formula. We first provide an algorithm dealing with Büchi automata, hence dealing with words indexed by the natural numbers. We then lay out a strategy to extend this result to automata on linear orderings.

The content of Chapters 3, 5, and 6 consists mainly of previously published work [11, 12, 13, 14]. These parts are explicitly identified and [highlighted](#).

Chapter 1

Preliminaries

In this chapter, we introduce the central definitions and notations we use to present our results, and the development of their proofs. We start by introducing some fundamental concepts of set theory, before turning to logic, where we focus on the basics of first-order and second-order logic and provide elements of terminology. We then turn to the theory of order, introducing the relevant notions to characterize linearly ordered sets. Finally, we present a notion of words indexed by linearly ordered sets, and a form of automata designed to read such words.

1.1 Logic

1.1.1 Sets, Relations, and Functions

Following the Zermelo-Fraenkel approach on set theory [50], we define a *class* simply as a (possibly infinite) collection of elements. A *set* is a special kind of class that is, intuitively, simple enough. A set can be infinite, for instance the class of all natural numbers $\{0, 1, 2, 3, \dots\}$ denoted by $\mathbb{N}$ is a set. However, some classes cannot be considered as sets without causing paradoxes (such as Russell's paradox [101], which involves the set of all sets that do not contain themselves). Without going into further detail, simply assume that we forbid ourselves to consider any collection of classes.

Given a set S and an element e , the fact that e belongs to S is denoted by $e \in S$, and the opposite by $e \notin S$. A set that contains a single element is called a *singleton*. There exists a unique set that does not contain any element, which is called the *empty set*, and is denoted by $\emptyset$. The *Cartesian product* of $n \in \mathbb{N}$ sets $S_1, \dots, S_n$ is denoted by $S_1 \times \dots \times S_n$, and is defined as the set of n -tuples $\{(s_1, \dots, s_n) \mid s_i \in S_i, \text{ for all } i \in [1, n]\}$. The Cartesian product of n copies of a set S is denoted by S^n . A *relation* over $S_1 \times \dots \times S_n$ is simply a subset of $S_1 \times \dots \times S_n$. For any $n \in \mathbb{N}$, a *function* from $D_1 \times \dots \times D_n$ (its *domain*) to a set C (its *codomain*) is a mapping that assigns to each element in $D_1 \times \dots \times D_n$ a unique element in C . The positive integer n is called the *arity* of

the relation or function. A function of arity zero is also called a *constant*. The terms *unary* and *monadic* refer to relations and functions of arity one, and the term *binary* refers to relations and functions of arity two. Notice that a relation can equivalently be defined as a function from $S_1 \times \cdots \times S_n$ to $\{\top, \perp\}$, where $\top$ and $\perp$ respectively represent the truth values *true* and *false*. Reciprocally, a function f of arity n can also be viewed as a relation $\mathcal{R}$ of arity $n + 1$, where $f(d_1, \dots, d_n) = c$ if and only if $(d_1, \dots, d_n, c) \in \mathcal{R}$.

For a binary relation $\mathcal{R}$, we may write $a\mathcal{R}b$ in place of $(a, b) \in \mathcal{R}$, and $a\not\mathcal{R}b$ in place of $(a, b) \notin \mathcal{R}$. Let $\mathcal{R}$ be a binary relation defined over S , then:

- $\mathcal{R}$ is *reflexive* if for all $a \in S$, one has $a\mathcal{R}a$;
- $\mathcal{R}$ is *irreflexive* if for all $a \in S$, one has $a\not\mathcal{R}a$;
- $\mathcal{R}$ is *symmetric* if for all $a, b \in S$, $a\mathcal{R}b$ iff $b\mathcal{R}a$;
- $\mathcal{R}$ is *asymmetric* if for all $a, b \in S$, if $a\mathcal{R}b$ then $b\not\mathcal{R}a$;
- $\mathcal{R}$ is *antisymmetric* if for all $a, b \in S$, $a\mathcal{R}b$ and $b\mathcal{R}a$ together imply $a = b$;
- $\mathcal{R}$ is *transitive* if for all $a, b, c \in S$, $a\mathcal{R}b$ and $b\mathcal{R}c$ together imply $a\mathcal{R}c$;
- $\mathcal{R}$ is *total* (or *connected*) if for all $a, b \in S$ with $a \neq b$, either $a\mathcal{R}b$ or $b\mathcal{R}a$.

A binary relation that is total, transitive, reflexive, and antisymmetric is called a *non-strict total order relation*, usually denoted by the symbol $\leq$. A binary relation that is total, transitive, irreflexive, and asymmetric is called a *strict total order relation*, usually denoted by the symbol $<$. One can always easily obtain a non-strict order from a strict order, and vice versa, by simply adjusting the behavior of the relation on all pairs of identical objects. In the following, we will assume that all the orders we consider are total and strict, although we may not always state it explicitly. The restriction to strict order is not constraining since one can easily adapt all our results to the case of non-strict orders. A *well-order* is an order relation over a set S such that every non-empty subset S' of S admits a least element, that is, there exists an element $e_{\min} \in S'$ such that for all elements $e \in S'$ one has $e_{\min} \leq e$. The standard order relation over the set of natural numbers $\mathbb{N}$ — i.e., $0 < 1 < 2 < \cdots$ — is a well-order. On the other hand, the standard order relation over the set of integer numbers $\mathbb{Z}$ — i.e., $\cdots < -1 < 0 < 1 < \cdots$ — is not a well-order, since any subset of $\mathbb{Z}$ containing infinitely many negative integers (including $\mathbb{Z}$ itself) does not admit a least element.

A function f is *injective* if every element y in its codomain admits at most one element x in its domain such that $f(x) = y$. A function f is *surjective* if every element y in its codomain admits at least one element x in its domain such that $f(x) = y$. A function that is both injective and surjective is called a *bijection*.

A set S is *countable* if it is either finite, or there exists a bijection between S and the set $\mathbb{N}$ of natural numbers. Otherwise, S is said to be *uncountable*. For instance, the set of integer numbers $\mathbb{Z}$ and the set of rational numbers $\mathbb{Q}$ are countable, while the set of real numbers $\mathbb{R}$ is uncountable.

Notations Throughout this thesis, we will need to consider sets, sets of sets, sets of sets of sets ... and so on, which can make the notations somewhat heavy. To ease the reading, let us now give some foresight on our choice of notations. We rely on two mechanisms to denote that an object is a set. Either we go from lower case to upper case symbols (e.g., a set P containing the elements $p, q, r \dots$), or we introduce a horizontal bar on top of the symbol (e.g., a set $\bar{p}$ containing the elements $\bar{p}, \bar{q}, \bar{r} \dots$). The latter will be favored when the set is not only considered as a set, but also as a singular element of a more complex structure.

1.1.2 Logic Terminology

Let us introduce the vocabulary related to logic that we will use throughout this thesis. We refer to e.g., [48] for a general and more thorough introduction to first-order and second-order logics.

A *formal language* is a language characterized by a *syntax* (the rules for building sentences) and a *semantic* (the meaning of symbols appearing in a sentence and their combinations). The syntactic rules of sentences in a formal language are usually very strict, with the benefit of allowing for a precise and unambiguous semantic, but comes with the tradeoff that it is usually much less expressive than natural languages, such as English.

We mostly focus on first-order and second-order logics. Intuitively, this means that the languages we consider only involve simple objects, mostly consisting of elements (usually numbers), sets of elements, relations and functions only involving primitive elements. Higher-order objects, such as functions that take another function as an argument for instance, are left out.

In this context, the *symbols* associated with such a language mainly comprise:

- the truth values $\top$ and $\perp$;
- first-order variables, denoted by lower case letters $x, y, z, \dots$, that range over primitive elements;
- second-order variables, denoted by upper case letters $P, R, X, \dots$, that range over sets of primitive elements;
- the logical connectives $\wedge$ (*and*), $\vee$ (*or*), $\neg$ (*negation*);
- the universal (resp. existential) quantifier $\forall$ (resp. $\exists$).

Additional symbols for various relations and functions may also appear, such as equality $=$, or an order relation $<$ for instance. The parentheses symbols ‘(’ and ‘)’ are also used to precisely establish the scope of quantification, and the precedence of the logical connectives.

The *signature* of a formal language is the description of all the non-logical symbols involved in this language, i.e., all the symbols that can appear in a formula, beyond the quantifiers, the variables, $\top$, $\perp$, and the logical connectives. In this work, these symbols correspond to either functions or relations, such as

$=$, $<$, $+$, or $\sqrt{\cdot}$ for instance. The signature also sets the expected arity of each relation and function symbol that it contains.

Given a signature σ , a σ -*structure* consists of a set — intended to be the domain $\mathbb{D}$ in which the first-order variables take their values — together with an explicit interpretation of each relation and function symbols appearing in σ , i.e., an assignment of concrete function or relation to each symbol in the signature σ . If a signature contains the relation symbol $\mathcal{R}$ or the function symbol f , we may use the same symbol to represent its interpretation if there is no ambiguity, just as we intend to use the symbol $<$ to denote both the symbol in a signature, and its actual interpretation as an order relation. Throughout this thesis, we always assume that the signature implicitly contains the symbol $=$ for equality, always interpreted as the binary relation of $\mathbb{D} \times \mathbb{D}$ that consists of every pair of identical elements.

A *predicate symbol* (or simply *predicate*) is a symbol in a signature that is meant to represent a relation. Some logics that we consider, which lay in between first-order and second-order, involve the notion of *unary uninterpreted predicate*. The presence of such a predicate is actually equivalent to the occurrence of a second-order variable, since a unary predicate corresponds to a relation of arity one (i.e., a subset of the domain), and the uninterpreted part refers to the fact that a structure is not required to explicitly define the relation associated with this symbol, just as for variables.

A *term* is inductively defined as follows. A first-order variable symbol is a term, and $f(t_1, \dots, t_n)$ is a term, where f is a function (symbol) of arity $n \in \mathbb{N}$, and $t_1, \dots, t_n$ are all terms themselves. An *atom* (or *atomic formula*) is either $\top$, $\perp$, $t_1 = t_2$, $P(t_1)$, $\mathcal{R}(t_1, \dots, t_n)$, where $t_1, \dots, t_n$ are terms, P is a second-order variable symbol, and where $\mathcal{R}$ is a relation (symbol) of arity n . A *formula* is inductively defined as follows. Every atom is a formula, and if φ and ψ are formulas, then $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\forall x. \varphi$, $\exists x. \varphi$, $\forall P. \varphi$, and $\exists P. \varphi$ are also formulas. The *quantifier depth* of a formula φ is equal to its number of nested quantifiers. In this work, we only rely on the notion of quantifier depth (in all technical developments) for formulas that do not involve second-order quantification. If φ is a first-order formula in *prenex normal form*, in other words of the form

$$Q_1 x_{i_1} Q_2 x_{i_2} \cdots Q_k x_{i_k} \psi(x_1, x_2, \dots, x_n, P_1, P_2, \dots, P_m),$$

where ψ is quantifier-free and $Q_i \in \{\exists, \forall\}$ for all $i \in [1, k]$, then its quantifier depth is k . In settings where second-order quantification is involved, it may be relevant to adapt this definition to account for alternation of first-order and second-order quantification. However, one can usually promote a first-order quantifier into a second-order one by enforcing that the quantified set is a singleton (as it is done in [114] for instance), making the distinction between first-order and second-order quantifiers superficial.

An occurrence of a variable x (resp. P) appearing in a formula φ is *bounded* if it is under the scope of a quantifier, i.e., if there exists a subformula of φ of the form $\forall x. \psi$ or $\exists x. \psi$ (resp. $\forall P. \psi$ or $\exists P. \psi$) such that this occurrence of x (resp. P) appears in ψ . Conversely, an occurrence of a variable x or P appearing in

a formula φ is said to be *free* if it is not under the scope of a quantifier, i.e., if it is not bounded. A formula without any free variable occurrence is referred to as a *sentence*. A variable is free in a formula φ if there is at least one free occurrence of this variable in φ , otherwise we say that the variable is bounded in φ . A formula φ whose set of free variables is $(x_1, \dots, x_n, P_1, \dots, P_m)$ may be written as $\varphi(x_1, \dots, x_n, P_1, \dots, P_m)$.

An *interpretation* I of a formula $\varphi(x_1, \dots, x_n, P_1, \dots, P_m)$ over a given structure $(\mathbb{D}, \mathcal{R}_1, \dots, \mathcal{R}_k, f_1, \dots, f_\ell)$ is an assignment of values $I(x_i) \in \mathbb{D}$ for all $i \in [1, n]$, and $I(P_i) \subseteq \mathbb{D}$ for all $j \in [1, m]$ to its free variables. Following the usual convention in logic, we do not allow to define an interpretation over the empty domain. By extension, an interpretation I assigns a value in $\mathbb{D}$ to each term appearing in φ , and a truth value to each subformula of φ , including φ itself. When an interpretation I makes a formula φ true, we say that I *models* φ , denoted by $I \models \varphi$. The semantics of an interpretation I is defined as follows:

- $I \models \top$ and $I \not\models \perp$ for all I ;
- $I \models t_1 = t_2$ iff $I(t_1) = I(t_2)$;
- $I \models P(t_1)$ iff $I(t_1) \in I(P)$;
- $I \models \mathcal{R}_i(t_1, \dots, t_j)$ iff $(I(t_1), \dots, I(t_j)) \in \mathcal{R}_i$;
- $I \models \neg\varphi$ iff $I \not\models \varphi$;
- $I \models \varphi \wedge \psi$ iff $I \models \varphi$ and $I \models \psi$;
- $I \models \varphi \vee \psi$ iff $I \models \varphi$ or $I \models \psi$ (it can be both);
- $I \models \exists x_i. \varphi(x_1, \dots, x_n, P_1, \dots, P_m)$ iff there exists $d \in \mathbb{D}$ such that $I_{[x_i=d]} \models \varphi$;
- $I \models \forall x_i. \varphi(x_1, \dots, x_n, P_1, \dots, P_m)$ iff for every $d \in \mathbb{D}$, one has $I_{[x_i=d]} \models \varphi$;
- $I \models \exists P_i. \varphi(x_1, \dots, x_n, P_1, \dots, P_m)$ iff there exists $D \subseteq \mathbb{D}$ such that $I_{[P_i=D]} \models \varphi$;
- $I \models \forall P_i. \varphi(x_1, \dots, x_n, P_1, \dots, P_m)$ iff for every $D \subseteq \mathbb{D}$, one has $I_{[P_i=D]} \models \varphi$;

where, for any term of the form $f_i(t_1, \dots, t_j)$, we have $I(f_i(t_1, \dots, t_j)) = f_i(I(t_1), \dots, I(t_j))$. Notice that $\mathcal{R}_i$ and f_i act as both the relation and function symbols from the signature (when they occur in a formula), and the concrete interpretation of these symbols by the structure¹. In the last four rules, the notation $I_{[x_i=d]}$ (resp. $I_{[P_i=D]}$) denotes an extension² of the interpretation I to one additional first-order variable x (resp. second-order variable P_i) that takes the value d (resp. D), i.e., the interpretation defined as $I_{[x_i=d]}(x_j) = I(x_j)$ for all $j \in [1, n]$ with $j \neq i$, $I_{[x_i=d]}(x_i) = d$, and $I_{[x_i=d]}(P_j) = I(P_j)$ for all $j \in [1, m]$

¹This is actually also the case for '=' in the second rule.

²When blocks of quantification over the same variable occur, e.g., $\exists x_i \forall x_i$, then $I_{[x_i=d]}$ is not an extension, but a substitution of I where the value attributed to x_i is overwritten.

(resp. $I_{[P_i=D]}(x_j) = I(x_j)$ for all $j \in [1, n]$, $I_{[P_i=D]}(P_j) = I(P_j)$ for all $j \in [1, m]$ such that $j \neq i$, and $I_{[P_i=D]}(P_i) = D$).

A *model* of a formula φ is an interpretation that makes φ true, i.e., such that $I \models \varphi$ holds. A formula is *satisfiable* if it admits at least one model. A formula is *valid* (or is a *tautology*) if all its interpretations are models, or equivalently, if its negation is unsatisfiable.

For a given signature σ , a *theory* is a class of σ -structures. It can be (and often is) a singleton. Alternatively, a theory is also sometimes defined as the class of σ -structures that models a set of *axiomatic formulas* (or simply *axioms*), i.e., a (possibly infinite) set of sentences inherently assumed to be true. In the latter case, a formula is satisfiable in the theory if there exists a model defined over a structure that satisfies the axioms. Such theories are referred to as *axiomatic theories*. These two definitions for a theory are close, but not exactly equivalent. If one can easily define the class of structures that models a given set of axioms, there may not exist a set of axioms that characterizes exactly a given class of structures. The Löwenheim-Skolem theorem [109] actually states that if a first-order theory characterized by a set of axioms admits a model of infinite cardinality, then it also admits model of any infinite cardinality. Intuitively this means that if one tries to axiomatize the set of natural numbers $\mathbb{N}$ for instance, then these axioms would also admit a model of cardinality strictly greater than that of $\mathbb{N}$. (The notion of cardinality will be formally introduced in Section 1.2.3).

A theory is *decidable* for satisfiability (resp. validity) if there exists an algorithm that can establish, for any given formula of this theory, whether it is satisfiable (resp. valid) or not. Notice that if the language of formulas is closed under negation (i.e., the negation of a formula is still a formula), deciding satisfiability is equivalent to deciding validity.

An axiomatic theory is *complete* if for any sentence φ in that theory, it is possible to deduce from the axioms either the sentence φ , or its negation $\neg\varphi$. In other words, in an incomplete theory, some statements are neither true nor false, but simply independent from the theory. (A famous example is the axiom of choice in the Zermelo-Fraenkel fragment of set theory [50].) An axiomatic theory is *consistent* if for any sentence deduced from the axioms, it is not possible to also deduce its negation. We will not discuss the notions of completeness and consistency of a theory defined as a class of structures (the former is not very natural, and the latter simply is trivial).

A theory T_1 is *at least as expressive as* another theory T_2 if, for every formula φ_2 of T_2 , there exists a formula φ_1 of T_1 such that the set of models of φ_1 coincides with the set of models of φ_2 . Two theories are *equivalent* if they are both at least as expressive as each other. Two theories are *incomparable* if both admit at least one formula whose set of models does not coincide with the set of models of any formula of the other theory.

We denote most theories by a combination of a prefix denoting the order of the theory, together with its signature. The prefix can either be FO, MSO, or MFO, respectively corresponding to the following:

- First-Order (FO): Formulas can only involve first-order variables, quantified or not;
- Monadic³ Second-Order (MSO): Formulas can involve both first-order and second-order variables, without any restrictions on their quantification;
- Monadic First-Order (MFO): First-order variables can appear and be quantified in the formulas, but second-order variables can only appear unquantified.

We also consider a variant of the monadic first-order case, where the number of distinct second-order variable symbols appearing in a formula is bounded by a fixed positive integer n . The associated prefix is then $M_n\text{FO}$. This restriction of the monadic first-order frame (which can also be seen as an extension of the pure first-order one) is often described in the literature as the possibility to involve up to n unary uninterpreted predicates.

Later on (particularly in Chapter 2), other forms of logics will be mentioned which do not exactly fit into this framework, such as temporal logics for instance. Their specificities will be introduced in the relevant sections. Formulas in such logics can usually be equivalently expressed in a logic that fits the definitions introduced in this section.

Syntactic Sugar Following the strict syntax we described can make the formulas overly intricate, and therefore hard to read. For this reason, we often resort to some syntactic sugar when writing explicit formulas, such as implications of the form $\varphi \Rightarrow \psi$ and equivalences of the form $\varphi \Leftrightarrow \psi$, which respectively stand for $\neg\varphi \vee \psi$ and $(\varphi \wedge \psi) \vee ((\neg\varphi) \wedge (\neg\psi))$. The conjunction (resp. disjunction) operator $\wedge$ (resp. $\vee$) is associative, which allows to discard parentheses in sequences of conjunctions (resp. disjunctions). Finally, when we write e.g., $\exists x. \varphi$, the scope of the quantifier ranges over the entire formula φ . If we need to restrict this range, we use parentheses, e.g., $(\exists x. \varphi) \wedge \psi$.

Example 1. Consider the formula $\varphi_1 = (\forall P. (\exists x. P(x)) \Rightarrow P(x)) \wedge \forall x. P(x)$. This formula only involves the two variable symbols x and P . However, the various occurrences of these variables are very different. The occurrences of x and P in blue, red, and purple are all bounded, while their occurrences in orange are free. The formula φ_1 therefore possesses one free first-order variable and one free second-order variable, and would therefore be denoted as $\varphi_1(x, P)$.

This formula expresses two things. First, that the element denoted by x belongs to every non-empty subset of the domain. Second, that the set denoted by P coincides with the entire domain. The formula φ_1 could be rewritten with less ambiguity as $(\forall R. (\exists y. R(y)) \Rightarrow R(x)) \wedge \forall z. P(z)$, so as to make every occurrence of x and P free.

³In plain second-order logic, one can quantify over functions and relations of any arity. The ‘monadic’ keyword conveys the idea that only unary uninterpreted relations (or equivalently second-order variables) can be considered in this setting.

The formula $\varphi_1(x, P)$ is satisfiable over any structure in which the domain is a singleton, and unsatisfiable over any domain of another cardinality. Indeed, the subformula $\forall R. (\exists y. R(y)) \Rightarrow R(x)$ necessarily evaluates to false if the domain contains two or more elements. Let $\mathbb{D} = \{d\}$ be any singleton domain. An interpretation that assigns d to x , and $\{d\}$ to P is necessarily a model of $\varphi_1(x, P)$. The formula $\varphi_1(x, P)$ is however not valid. For instance, over a similar domain $\mathbb{D} = \{d\}$, the interpretation that assigns d to x , and the empty set $\emptyset$ to P is not a model of φ_1 .

Example 2. Presburger arithmetic [85, 60] is the first-order theory of natural numbers with addition, also denoted $\text{FO}(\mathbb{N}, <, 0, 1, +)$. More precisely, the domain of Presburger arithmetic is $\mathbb{N}$. Its signature solely consist of the constants 0 and 1 (although not necessary⁴), the usual order relation over the natural numbers, addition (the binary function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$ that matches the usual addition operator over natural numbers), and equality (which is implicit as explained before). Furthermore, formulas can only involve first-order variables. This theory can equivalently be characterized by a set of first-order axioms. Presburger arithmetic is decidable, complete, and consistent [85].

The extension of Presburger arithmetic with multiplication corresponds to Peano arithmetic [68], which is strictly more expressive than Presburger arithmetic, and is actually undecidable and incomplete [55].

1.2 Theory of Order

1.2.1 Linear Orderings

We first give basic definitions and results about linear orderings, and refer to [100] for further details.

A *linear ordering* J is a totally ordered set, i.e., a set equipped with an (strict and total) order relation $<_J$. Two linear orderings J and K — respectively associated with the order relations $<_J$ and $<_K$ — are *order-isomorphic* if there exists an order-preserving bijection between J and K . Formally, let b be such a bijection (from J to K), then $b(j_1) <_K b(j_2)$ iff $j_1 <_J j_2$ for all $j_1, j_2 \in J$. We denote by $-J$ the *backwards* linear ordering that corresponds to J with its ordering reversed, that is, the ordering $<_{-J}$ defined as $j_2 <_{-J} j_1$ if and only if $j_1 <_J j_2$, for all $j_1, j_2 \in J$. The class of all linear orderings is denoted by $\mathcal{L}$.

The *order type* of a linear ordering J is the class of all linear orderings that are order-isomorphic to J . The order types of a singleton, the set $\{1, 2, \dots, n\}$, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ are respectively denoted by 1, n , ω , ζ , η , and λ . Notice that the order type of any non-empty open interval of $\mathbb{R}$ is λ , and that the order type of any non-empty open interval of $\mathbb{Q}$ is η . For convenience, and when the underlying ordering is clear, we sometime substitute a linear ordering with its corresponding set (in particular for $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$).

⁴The constant 0 can be defined as the natural number satisfying $n + 0 = n$ for all $n \in \mathbb{N}$, and 1 can be defined as the immediate successor of 0.

An *ordinal number* (or simply *ordinal*) is the order type of a linear ordering J for which the associated order relation $<_J$ is a well-order. Another way to present it is to say that J does not admit a subset of order type $-\omega$. A more in-depth presentation of the ordinals is given in Section 1.2.2.

In the following we will often abuse the notations by assimilating a linear ordering with its order type. The reason for it is that we are usually not interested in the nature of the elements themselves, but in their underlying order relation.

The concatenation of two linear orderings J and K (with respective order relations $<_J$ and $<_K$) is denoted by $J+K$. It corresponds to the linear ordering composed of the set of pairs $\{(j, 1) \mid j \in J\} \cup \{(k, 2) \mid k \in K\}$, equipped with the order relation $<$ defined by $(j_1, 1) < (j_2, 1)$ if $j_1 <_J j_2$, $(k_1, 2) < (k_2, 2)$ if $k_1 <_K k_2$, and $(j, 1) < (k, 2)$ for every $j \in J$ and $k \in K$.

More generally, let K and J_k for all $k \in K$ be linear orderings. The linear ordering $\Sigma_{k \in K} J_k$ is obtained by concatenating the orderings J_k w.r.t. the ordering K . Formally, $\Sigma_{k \in K} J_k$ is the linear ordering defined over the set of pairs (j, k) such that $j \in J_k$ and $k \in K$, with the order relation $(j_1, k_1) < (j_2, k_2)$ iff either $k_1 <_K k_2$, or $k_1 = k_2$ and $j_1 <_{J_{k_1}} j_2$. When $J_k = J$ for every $k \in K$, we may write $J \cdot K$ in place of $\Sigma_{k \in K} J_k$. These operators naturally extend to order types. For instance, the order type $\omega \cdot \omega$ (also written as ω^2) is the class of all linear orderings that are order-isomorphic to the linear ordering⁵ $\mathbb{N} \cdot \mathbb{N}$, which corresponds to the set⁶ $\mathbb{N} \times \mathbb{N}$ equipped with the ordering $<$ defined as $(n_1, m_1) < (n_2, m_2)$ if and only if $n_1 <_{\mathbb{N}} n_2$, or $n_1 = n_2$ and $m_1 <_{\mathbb{N}} m_2$, for all $n_1, n_2, m_1, m_2 \in \mathbb{N}$.

A *cut* of a linear ordering J is a partition of J into a pair (K, L) of sets such that for every $(k, \ell) \in K \times L$, one has $k < \ell$. The linear ordering consisting of the set of all cuts of an ordering J equipped with the order relation $(K_1, L_1) < (K_2, L_2)$ iff $K_1 \subsetneq K_2$ is denoted by $\hat{J}$. The ordering $\hat{J}$ always has as its first element $(\emptyset, J)$ and as its last element $(J, \emptyset)$. These first and last cuts are sometimes ignored. We use the notation $\hat{J}^*$ to denote the ordering $\hat{J}$ deprived of the first and the last cuts, i.e., $\hat{J}^* = \hat{J} \setminus \{(\emptyset, J), (J, \emptyset)\}$. For every $j \in J$, the *consecutive* cuts $(\{k \mid k < j\}, \{ \ell \mid j \leq \ell \})$ and $(\{k \mid k \leq j\}, \{ \ell \mid j < \ell \})$ are respectively denoted by j^- and j^+ . Let $c = (K, L)$ be a cut that is neither the first nor the last one of J , i.e., $K \neq \emptyset$ and $L \neq \emptyset$. If K admits a greatest element and L admits a smallest element, then c is called a *jump*. If K does not admit a greatest element and L does not admit a smallest element, then c is called a *gap*.

A linear ordering J is (*Dedekind*) *complete* if the set $\hat{J}$ does not contain any gap. For instance, $\mathbb{N}$ and $\mathbb{R}$ are complete, while $\mathbb{Q}$ is incomplete. A linear ordering J is *dense* if between any two distinct elements lies another one, i.e., for every $j_1, j_2 \in J$ such that $j_1 <_J j_2$, there exists j_3 such that $j_1 <_J j_3 <_J j_2$. Notice that a linear ordering is dense iff it does not admit two consecutive elements. A linear ordering J is *scattered* if it does not contain a dense subordering.

⁵Here $\mathbb{N}$ is viewed as the ordering of the natural numbers.

⁶Here $\mathbb{N}$ only denotes the set of natural numbers, without its usual ordering $<_{\mathbb{N}}$.

Example 3. Consider the linear ordering $\mathbb{N} + \mathbb{Z}$ consisting in the set of natural numbers followed by the set of integers. A depiction of this set and its cuts is given in Figure 1.1. The elements of this ordering are represented as dots, and its cuts as vertical lines. Notice that this ordering admits a single gap (depicted in red), hence it is incomplete. It is also scattered.



Figure 1.1: The linear ordering $\mathbb{N} + \mathbb{Z}$.

1.2.2 Ordinal Numbers

Let us give more details on the specific subclass of linear orderings that are the ordinals. As stated in Section 1.2.1, we view an ordinal number simply as the order type of a linear ordering that is well-ordered (or *well-ordering*). A *successor ordinal* is an the order type of a well-ordering that possesses a greatest element. A *limit ordinal* is the order type of a non-empty well-ordering that does not possess a greatest element. An ordinal is either the first ordinal 0 (zero), which corresponds to the empty set, a successor ordinal, or a limit ordinal. The class of all ordinals is denoted by $\mathcal{O}$. The first infinite ordinal is denoted by ω (the order type of $\mathbb{N}$), and the least uncountable ordinal is denoted by ω_1 .

Other equivalent definitions of ordinals appear in the literature. For instance, an ordinal can be defined as the set of all lesser ordinals, with the first ordinal being the empty set, for every $n \in \mathbb{N}$, the n -th ordinal corresponding to the set of the n first ordinals, the ordinal ω corresponding to the set of all finite ordinals, and so on. We refer to the chapter dedicated to ordinals in [100] for a general introduction to ordinal numbers.

1.2.3 Cardinal Numbers

The *cardinality* of a set relates to the number of elements it contains. This number can be infinite, and multiple distinct infinite cardinalities exist. More precisely, the cardinality of a set S is defined as the equivalence class of all the sets S' such that there exists a bijection between S and S' , and is denoted by $\text{Card}(S)$. Such classes are usually represented by a specific symbol which is called a *cardinal number*. For the cardinalities of finite sets we simply use the natural numbers. The smallest infinite cardinal number is $\aleph_0$, and corresponds to the cardinality of the set of natural numbers $\mathbb{N}$. The consecutive cardinal numbers are denoted $\aleph_1, \aleph_2, \dots$. The least ordinal of its class of cardinality is called an *initial ordinal*. For instance, ω is the initial ordinal associated with $\aleph_0$, and ω_1 is the initial ordinal associated with $\aleph_1$.

The *continuum hypothesis* postulate that the set of real numbers $\mathbb{R}$ has the cardinality $\aleph_1$, or equivalently, that there does not exist a set that has a

cardinality strictly between that of $\mathbb{N}$ and that of $\mathbb{R}$. The continuum hypothesis has been shown to be independent from ZFC [56, 35], i.e., the Zermelo-Fraenkel fragment of set theory with the axiom of choice.

1.2.4 Cofinality and Coinitiality

Given a linearly ordered set S , a subset $S' \subseteq S$ is *cofinal* in S if for every element $s \in S$, there exists $s' \in S'$ such that $s \leq s'$. Analogously, a subset S' is *coinital* in S if for every element $s \in S$, there exists $s' \in S'$ such that $s \geq s'$. The *cofinality* (resp. *coinitality*) of a linearly ordered set S is the least cardinality of the cofinal (resp. coinital) subsets of S . For convenience, we may replace the cofinality with the initial ordinal number associated with it, and the coinitality with the reverse of the initial ordinal associated with it. For instance we may say that the cofinality of $\mathbb{N}$ is ω , that the coinitality of $\mathbb{Z}$ is $-\omega$, and that the cofinality of ω_1 is ω_1 itself, in place of respectively $\aleph_0$, $\aleph_0$, and $\aleph_1$.

To illustrate these concepts, let us consider the following result (excerpted from [100], Exercise 3.26).

Property 1.2.1. *There does not exist a subset of $\mathbb{R}$ that is order-isomorphic to ω_1 .*

Proof. By contradiction, let us assume that such a subset of the reals exists, and let us call it R . The cofinality of R must be the same as the one of ω_1 (i.e., ω_1 itself) which is strictly greater than ω . This first implies that R cannot admit a maximum element, otherwise it would have cofinality 1. Let $M = \sup\{r \mid r \in R\}$ (we allow $M = +\infty$), where $\sup$ denotes the supremum of a set, i.e., its least upper bound. Let $(c_i)_{i \in \mathbb{N}}$ be an infinite sequence of strictly increasing real numbers converging towards M . For all $i \in \mathbb{N}$, we define r_i as the smallest element of R that is greater or equal to c_i . This smallest element always exists because, since R is of order type ω_1 , R is necessarily well ordered. Therefore, if we consider the subset of R comprised of the elements that are greater or equal to c_i , this subset admits a smallest element.

It follows that the set $\{r_i \mid i \in \mathbb{N}\}$ of size ω is cofinal in R , which contradicts the fact that the cofinality of R is strictly greater than ω . $\square$

1.3 Languages, Regular Expressions, and Automata

1.3.1 Words Indexed by Linear Orderings and Regular Expressions

We now define the notions of words and regular expressions on linear orderings, as introduced in [20, 8].

A *word indexed by a linear ordering* is a totally ordered sequence of letters. Formally, given an alphabet Σ and a linear ordering J , a word $u = (\alpha_j)_{j \in J}$ indexed by J is a mapping $u : J \rightarrow \Sigma$. The *length* $|u|$ of u is J itself. The *empty*

word ε denotes the word indexed by the empty set. Two words $u_1 = (\alpha_j)_{j \in J}$ and $u_2 = (\beta_k)_{k \in K}$ are considered to be equal if there exists an order-preserving bijection b from J to K such that for all $j \in J$, $\alpha_j = \beta_{b(j)}$. Informally, for a given word u , only the order type of its underlying linear ordering $|u|$ is relevant, and not this linear ordering itself. In this context, a *language* is simply a class of words.

We now define a product operator on words. Let K and J_k for all $k \in K$ be linear orderings. Let $u_k = (\alpha_{j,k})_{j \in J_k}$ be a word of length J_k , for every $k \in K$. The *product* $\prod_{k \in K} u_k$ is the word $u = (\alpha_{j,k})_{(j,k) \in |u|}$ of length $|u| = \sum_{k \in K} J_k$. The product of two words u_1 and u_2 is denoted by $u_1 \cdot u_2$.

Let u be a word indexed by a linear ordering. Given two cuts $b, c \in \widehat{|u|}$, the *segment* of u between b and c , denoted $u_{[b,c]}$, is the word indexed by the subordering $J_{[b,c]}$ of J corresponding to the interval $\{j \in |u| \mid b \leq j^- \wedge j^+ \leq c\}$, such that for all $j \in J_{[b,c]}$, $u_{[b,c]}(j) = u(j)$.

A generalized notion of regular expressions has been introduced in [20, 8] to describe languages of words indexed by linear orderings. Let X and Y be two classes of words on linear orderings. We define the following operators:

- *Union*: $X + Y = \{z \mid z \in X \cup Y\}$;
- *Concatenation*: $X \cdot Y = \{x \cdot y \mid x \in X, y \in Y\}$;
- *Finite repetition*: $X^* = \{\prod_{i \in [1,n]} x_i \mid n \in \mathbb{N} \wedge \forall i \in [1,n]. x_i \in X\} \cup \{\varepsilon\}$;
- *Infinite repetition*: $X^{+\omega} = \{\prod_{i \in \mathbb{N}} x_i \mid \forall i \in \mathbb{N}. x_i \in X\}$;
- *Reverse infinite repetition*: $X^{-\omega} = \{\prod_{i \in (-\mathbb{N})} x_i \mid \forall i \in -\mathbb{N}. x_i \in X\}$;
- *Ordinal repetition*: $X^{+\#} = \{\prod_{i \in o} x_i \mid o \in \mathcal{O} \wedge \forall i \in o. x_i \in X\}$;
- *Reverse ordinal repetition*: $X^{-\#} = \{\prod_{i \in -o} x_i \mid o \in \mathcal{O} \wedge \forall i \in o. x_i \in X\}$;
- *Diamond operator*: $X \diamond Y = \{\prod_{j \in J \cup \widehat{J}^*} z_j \mid J \in \mathcal{L} \wedge \forall j \in J. z_j \in X \wedge \forall j \in \widehat{J}. z_j \in Y\}$.

The diamond ($\diamond$) operator can be understood as an iteration on all possible linear orderings. In settings where we only consider scattered, or countable linear orderings, we may adapt this operator to range over the appropriate subclass of linear orderings. We write $X^\diamond$ in place of $X \diamond \varepsilon$, and use the notation $\Sigma^\diamond$ to denote the class of all words indexed by a linear ordering over the alphabet Σ .

We also define a *shuffle* operator that generates density. Let $X_1, \dots, X_n \subseteq \Sigma^\diamond$, for some $n \geq 1$. We denote by $\text{sh}(X_1, \dots, X_n)$ the class of words of $\Sigma^\diamond$ that can be written as $\prod_{r \in R} x_r$ such that

- R is a non-empty, dense, and complete linear ordering without first or last element, and
- there exists a partition $\{R_1, \dots, R_n\}$ of R such that every R_i is dense in R , and for every $r \in R$ and $i \in [1, n]$, if $r \in R_i$ then $x_r \in X_i$.

Notice that the order type of the dense and complete ordering R mentioned in the shuffle definition is not necessarily λ . Furthermore, as shown in [8], although the shuffle operator relies on a complete ordering R , it is possible for a word $w \in \text{sh}(X_1, \dots, X_n)$ to be such that $|w|$ is incomplete. This occurs when there exists $i \in [1, n]$ such that $\varepsilon \in X_i$, and there exists $j \in [1, n]$ such that $X_j \neq \{\varepsilon\}$. Intuitively, every element of R_i associated with ε does not belong to $|w|$, but instead matches a gap in $\widehat{|w|}$.

A *regular expression* (or *rational expression*) is a finite representation of a language, and is inductively defined as follows. The symbols $\emptyset$, ε , and α for every letter $\alpha \in \Sigma$ are regular expressions, respectively describing the empty language $\emptyset$, the language containing only the empty word $\{\varepsilon\}$, and $\{\alpha\}$, the language containing only the single-letter word α . Any finite combination of regular expressions using the operators defined previously is also a regular expression.

1.3.2 Automata

Many forms of finite-state automata have been introduced, each dealing with words indexed by various classes of linear orderings. We recall the definitions of some classical ones, up to *automata on linear orderings* [20] which operate on words indexed by arbitrary linear orderings.

Finite-Word Automata The first form of automata we consider is automata that read finite words, i.e., words indexed by a finite ordering.

Definition 1.3.1. A (non-deterministic) finite-word automaton $\mathcal{A}$ is a tuple $(Q, \Sigma, I, F, \Delta)$, where Q is a finite set of states, Σ is a finite alphabet, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of accepting states, and $\Delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$ is a transition relation.

A transition labeled by the empty word ε , i.e., of the form $(q_1, \varepsilon, q_2) \in \Delta$, is called an ε -transition. If I is a singleton, and Δ can be expressed as a transition function δ from $Q \times \Sigma$ to Q — i.e., for any pair $(q, \alpha) \in Q \times \Sigma$, the transition relation Δ contains at most one triple of the form (q, α, q') — then $\mathcal{A}$ is a *deterministic* finite-word automaton.

A *path* π (sometimes also called a *run*) in a finite-word automaton $\mathcal{A} = (Q, \Sigma, I, F, \Delta)$ is defined as a finite sequence of transitions $((q_1, \alpha_1, q'_1), \dots, (q_n, \alpha_n, q'_n))$ of Δ such that for every $i \in [1, n-1]$, $q'_i = q_{i+1}$. We say the path π reads the word $\alpha_1 \cdot \dots \cdot \alpha_n$, or that the word $\alpha_1 \cdot \dots \cdot \alpha_n$ labels the path π . Such a path π is *accepting* if $q_1 \in I$ and $q'_n \in F$.

The language accepted by an automaton $\mathcal{A}$ is defined as the class of words labeling accepting paths in $\mathcal{A}$, and is denoted by $L(\mathcal{A})$. Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are *equivalent* if they accept the same language. Every non-deterministic finite-word automaton can be turned into an equivalent deterministic one [89], by a process called *determinization*.

A language of finite words is *regular* (or *rational*) if there exists a finite-word automaton $\mathcal{A}$ that accepts this language. A regular language of finite words can

always be described by a regular expression (as introduced in Section 1.3.1) that only involves the union, concatenation, and finite repetition operators, and reciprocally, any language described by such a restricted regular expression is accepted by a finite-word automaton [70, 79].

The *complement* of a language L defined over an alphabet Σ corresponds to the language $\Sigma^* \setminus L$, and is denoted L^C . Regular languages are closed under complementation, or in other words, for any finite-word automaton $\mathcal{A}$, there exists an automaton $\mathcal{A}^C$ such that $L(\mathcal{A}^C) = L(\mathcal{A})^C$.

Example 4. Consider the regular language represented by the regular expression $(abb)^* \cdot a + b(bb)^*$. This language is accepted by the non-deterministic finite-word automaton depicted in Figure 1.2. This automaton is defined as the tuple $(Q, \Sigma, I, F, \Delta)$, where the set of states Q is equal to $\{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$, the alphabet Σ is $\{a, b\}$, the set of initial states I is the singleton $\{q_1\}$, the set of accepting states is $\{q_5, q_6\}$, and the transition relation is

$$\Delta = \{(q_1, a, q_2), (q_2, b, q_3), (q_3, b, q_4), (q_4, a, q_5), (q_4, a, q_2), (q_1, b, q_6), (q_6, b, q_7), (q_7, b, q_6)\}.$$

In Figure 1.2 (as in all representations of automata in this thesis), initial states are represented as circles with an ingoing arrow without origin ($\rightarrow \bigcirc$), and accepting states as circles with a doubled edge ($\bigcirc \bigcirc$).

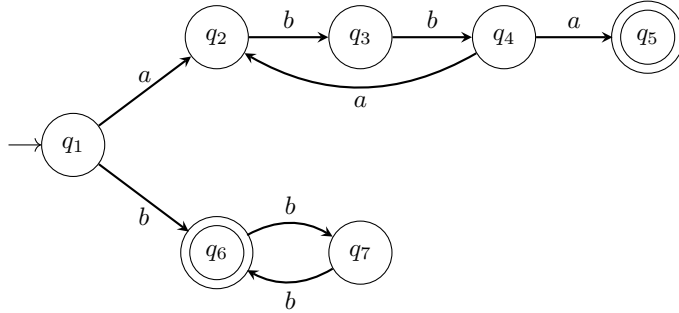


Figure 1.2: Finite-word automaton accepting $(abb)^* \cdot a + b(bb)^*$.

Infinite-Word Automata Let us now introduce automata that read infinite words, defined as words indexed by $\mathbb{N}$. We only focus on Büchi and Muller automata, but many other variants exist, such as Rabin automata, parity automata, Streett automata ... (see [113] for a survey).

Definition 1.3.2. A (non-deterministic) Büchi automaton $\mathcal{A}$ is a tuple $(Q, \Sigma, I, F, \Delta)$, where Q is a finite set of states, Σ is a finite alphabet, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of accepting states, and $\Delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$ is a transition relation.

Notice that the syntax of Büchi automata coincides with that of finite-word automata. The same applies for the deterministic version of Büchi automata. A *path* π in a Büchi automaton is an infinite sequence of transitions $((q_n, \alpha_n, q'_n))_{n \in \mathbb{N}}$ of Δ such that for every $n \in \mathbb{N}$, one has $q'_n = q_{n+1}$. Let $\text{inf}(\pi)$ denote the set of states visited infinitely often by π . More formally, we have

$$\text{inf}(\pi) = \{q \in Q \mid \forall m \in \mathbb{N}. \exists n > m. q_n = q\}$$

The word $\Pi_{n \in \mathbb{N}} \alpha_n$ indexed by $\mathbb{N}$ labels the path π . Such a path π is *accepting* if $\text{inf}(\pi) \cap F \neq \emptyset$, or in other words, if at least one accepting state is visited infinitely often by π .

Definition 1.3.3. A (non-deterministic) Muller automaton $\mathcal{A}$ is a tuple $(Q, \Sigma, I, \overline{F}, \Delta)$, where Q is a finite set of states, Σ is a finite alphabet, $I \subseteq Q$ is a set of initial states, $\overline{F} \subseteq 2^Q$ is a set of accepting components, and $\Delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times Q$ is a transition relation.

The key difference with finite-word and Büchi automata is the definition of the accepting components $\overline{F}$ which is not anymore a set of states, but a set of sets of states. *Deterministic* Muller automata are defined analogously to the Büchi and finite-word cases.

As in a Büchi automaton, a *path* π in a Muller automaton is an infinite sequence of transitions $((q_n, \alpha_n, q'_n))_{n \in \mathbb{N}}$ of Δ such that for every $n \in \mathbb{N}$, one has $q'_n = q_{n+1}$. However, a path π in a Muller automaton is *accepting* if $\text{inf}(\pi) \in \overline{F}$, or in other words, if the set of states visited infinitely often by π coincides exactly with a component in $\overline{F}$.

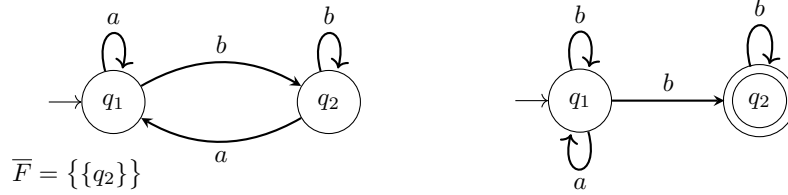
Non-deterministic Muller and Büchi automata share the same expressive power [78], and the languages that they accept are called ω -regular. However, while deterministic Muller automata are as expressive as their non-deterministic version⁷, the same is not true for Büchi automata. This is illustrated in Example 5, that shows an ω -regular language that cannot be accepted by any deterministic Büchi automaton.

Example 5. Consider the ω -regular language L described by the regular expression $(a + b)^* \cdot b^{+\omega}$, i.e., the set of infinite words over the alphabet $\{a, b\}$ which contains finitely many occurrences of the letter a . Figure 1.3a and Figure 1.3b respectively depict a deterministic Muller automaton and a non-deterministic Büchi automaton that accept L .

There does not exist a deterministic Büchi automaton that accept L . Assume by contradiction that such an automaton $\mathcal{A}_D$ exists, and let F be its set of accepting states. Let us show how to build an accepting path in $\mathcal{A}_D$ labeled by a word where the letter a occur infinitely often. Start with an accepting path π_0 labeled by $b^{+\omega} \in L$, and let $n_1 > 0$ be such that π_0 visits an accepting state after following exactly n_1 transitions. Then, consider an accepting path π_1 in $\mathcal{A}_D$

⁷For this reason, only deterministic Muller automata are typically considered. But in this work, their definition also serves as an intermediate step to introduce a more complex type of automata, for which their non-deterministic version is more suitable.

labeled by the word $b^{n_1} \cdot a \cdot b^{+\omega} \in L$. Since the automaton $\mathcal{A}_D$ is deterministic, the two paths π_0 and π_1 necessarily coincide on their shared prefix b^{n_1} . Let $n_2 > 0$ be such that the path π_1 visits an accepting state after reading exactly $n_1 + 1 + n_2$ letters. Repeating this construction $k > |F|$ times yields an accepting path π_k labeled by a word of the form $b^{n_1} \cdot a \cdot b^{n_2} \cdot a \cdot b^{n_3} \cdot a \cdots a \cdot b^{n_k} \cdot a \cdot b^{+\omega} \in L$, such that π_k visits an accepting state just before each occurrence of the letter a . This means that π_k visits accepting states on at least k occasions over the prefix $b^{n_1} \cdot a \cdot b^{n_2} \cdot a \cdot b^{n_3} \cdot a \cdots a \cdot b^{n_k} \cdot a$. Since $k > |F|$, it follows that π_k visits the same accepting state q_F on at least two distinct occasions. This implies the existence of a cycle involving an accepting state and a transition labeled by the letter a . Using this cycle, one can straightforwardly build an accepting run of $\mathcal{A}_D$ labeled by a word containing infinitely many occurrences of the letter a , hence the contradiction.



(a) Deterministic Muller automaton. (b) Non-deterministic Büchi automaton.

Figure 1.3: Two infinite-word automata accepting $(a + b)^* \cdot b^{+\omega}$.

Automata on Linear Orderings We now turn to automata on linear orderings, as introduced in [20]. They generalize the previously introduced notions of automata on finite and infinite words.

Definition 1.3.4. An automaton on linear orderings is a tuple $\mathcal{A} = (Q, \Sigma, I, F, \Delta^s, \Delta^\ell)$, where Q is a finite set of states, Σ is a finite alphabet, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of accepting states, $\Delta^s \subseteq Q \times \Sigma \times Q$ is a set of successor transitions, and $\Delta^\ell \subseteq (2^Q \times Q) \cup (Q \times 2^Q)$ is a set of (left and right) limit transitions.

The set Δ^s contains so called *successor transitions* of the form (q, α, q') , with $q, q' \in Q$ and $\alpha \in \Sigma$, alternatively written $q \xrightarrow{\alpha} q'$, which are similar to the transitions of finite-word and infinite-word automata. Although originally introduced without the possibility to use ε -transitions, automata on linear orderings can be extended quite naturally with ε -transitions, as shown in [10]. For the sake of simplicity, we will however not consider ε -transitions for automata on linear orderings in this work. The set Δ^ℓ comprises *left-limit transitions* of the form (P, q) , with $P \subseteq Q$ and $q \in Q$, written $P \rightarrow q$, and *right-limit transitions* of the form (q, P) , with $q \in Q$ and $P \subseteq Q$, written $q \rightarrow P$. We sometimes write sets of limit transitions in a more compact form, e.g., $p, q \rightarrow \{p, q\} \rightarrow p$ in place

of the set $\{p \rightarrow \{p, q\}, q \rightarrow \{p, q\}, \{p, q\} \rightarrow p\}$. The *size* of a limit transition $q \rightarrow P$ or $P \rightarrow q$ is equal to the cardinality $|P|$ of the set P .

We now define the notion of *path* in such an automaton. Let J be a linear ordering, and $\pi = (q_c)_{c \in \hat{J}}$ a word over the alphabet Q (i.e., each cut $c \in \hat{J}$ is associated with a state in the set Q). The *left-limit set* and *right-limit set* of π at c are the two subsets $\lim_c^- \pi$ and $\lim_c^+ \pi$ of Q defined as follows:

- $\lim_c^- \pi = \{q \in Q \mid \forall i < c. \exists k. i < k < c \wedge q = q_k\},$
- $\lim_c^+ \pi = \{q \in Q \mid \forall c < i. \exists k. c < k < i \wedge q = q_k\}.$

The left limit of π at c is therefore the set of states that are visited by π before c and infinitely close to c (or equivalently, in every left neighborhood of c). If c has a predecessor in the ordering $\hat{J}$, then this set is empty (intuitively it means that a *successor transition*, and not a *limit transition*, must be taken to reach the state mapped to c). The case of a right limit is handled symmetrically. For the first cut $c_{\min}$ and the last cut $c_{\max}$ of J , we deviate from the previous definition by setting $\lim_{c_{\min}}^- \pi = \emptyset$, and $\lim_{c_{\max}}^+ \pi = \emptyset$.

Definition 1.3.5. Let $\mathcal{A} = (Q, \Sigma, I, F, \Delta^s, \Delta^\ell)$ be an automaton on linear orderings and $w = (\alpha_j)_{j \in J}$ a word of length J . A path π labeled by w is a sequence of states $\pi = (q_c)_{c \in \hat{J}}$ of length $\hat{J}$ such that:

- For every $j \in J$, $(q_{j-}, \alpha_j, q_{j+})$ is a successor transition in Δ^s .
- For every cut $c \in \hat{J}$ that is not the first cut and does not have a predecessor in $\hat{J}$, $\lim_c^- \pi \rightarrow q_c$ is a left-limit transition in Δ^ℓ .
- For every cut $c \in \hat{J}$ that is not the last cut and does not have a successor in $\hat{J}$, $q_c \rightarrow \lim_c^+ \pi$ is a right-limit transition in Δ^ℓ .

A path is *accepting* if it starts in a state $p \in I$ and ends in a state $q \in F$. A word is accepted by $\mathcal{A}$ if it labels an accepting path. A path is *k-limited* if each limit transition $P \rightarrow q$ and $q \rightarrow P$ in the path is such that $|P| \leq k$. As shown in [20, 8, 10], the class of languages accepted by automata on linear orderings coincides exactly with the class of languages that can be represented by the (unrestricted) regular expressions introduced in Section 1.3.1.

Notice that a finite-word automaton can be seen as an automaton on linear orderings that has an empty set of limit transitions, i.e., such that $\Delta^\ell = \emptyset$. Infinite-word automata and automata on ordinal words [23, 32, 121] (which we do not explicitly define in this work) also correspond to particular cases of automata on linear orderings. Both correspond to automata with only left-limit transitions, with the additional constraint for the former that every left-limit transition leads to a unique accepting state that admits no outgoing transition, i.e., the automaton admits a unique accepting state q_F , every limit transitions is a left-limit transitions of the form $P \rightarrow q_F$, and the automaton does not contain successor transitions of the form (q_F, α, p) . For the case of Muller automata,

the translation is immediate, since every accepting component can straightforwardly be turned into a left-limit transition targeting a fresh unique accepting state, i.e., a fresh accepting state q_F is introduced, and every accepting component $P \in \bar{F}$ is turned into the left-limit transition $P \rightarrow q_F$. The translation of a Büchi automaton to an automaton on linear orderings can be done by generating a similar left-limit transition for every set of states that contains at least one accepting state of the original Büchi automaton, i.e., a fresh state q_F is introduced, for every subset $P \subseteq Q$ such that $P \cap F \neq \emptyset$ (where Q is the set of states of the input Büchi automaton, and F is its set of accepting states), one introduces the left-limit transition $P \rightarrow q_F$, and finally, q_F is set as the only accepting state.

Example 6. The automaton depicted in Figure 1.4 accepts $\{u\}$, where u is the word $u = a^{+\omega} \cdot b^{-\omega} \cdot b^{+\omega}$ defined over the alphabet $\{a, b\}$. Notice that u is indexed by the linear ordering $\mathbb{N} + \mathbb{Z}$ considered in Example 3. An accepting path can be described as follows: it maps the cuts (including the first one) of the first subordering $\mathbb{N}$ to the state q_1 , the gap to the state q_2 , the cuts of the subordering $\mathbb{Z}$ to the state q_3 , and the last cut to the state q_4 .

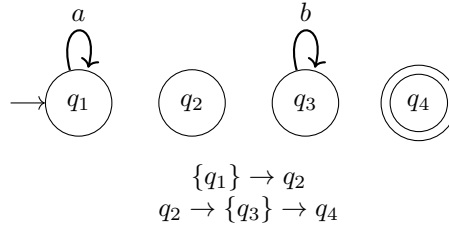


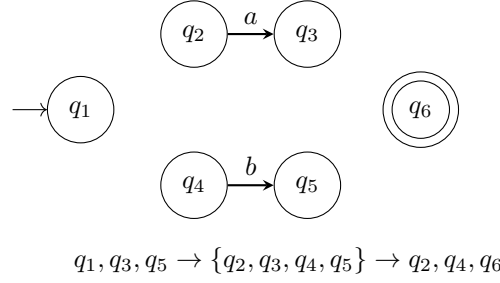
Figure 1.4: An automaton accepting $a^{+\omega} \cdot b^{-\omega} \cdot b^{+\omega}$.

Example 7. Consider the regular expression $\text{sh}(a, b)$. It describes the language containing every word over the alphabet $\{a, b\}$ indexed by a dense and complete ordering that does not admit a first or a last element, such that the distributions of letters a and b are both dense. The automaton depicted in Figure 1.5 accepts exactly this language.

Let us show that the word $w : \mathbb{R} \rightarrow \{a, b\}$ defined by $w(r) = a$ if $r \in \mathbb{Q}$, and $w(r) = b$ if $r \in \mathbb{R} \setminus \mathbb{Q}$, which belongs to the language associated with $\text{sh}(a, b)$, is accepted by this automaton. It suffices to consider the following accepting path π labeled by w :

- $\pi(-\infty) = q_1$,
- $\pi(+\infty) = q_6$,
- $\pi(r^-) = q_2$ and $\pi(r^+) = q_3$ for all $r \in \mathbb{Q}$, and
- $\pi(r^-) = q_4$ and $\pi(r^+) = q_5$ for all $r \in \mathbb{R} \setminus \mathbb{Q}$,

where $-\infty$ and $+\infty$ respectively denote the first and the last cut of $\mathbb{R}$.

Figure 1.5: An automaton accepting $\text{sh}(a, b)$.

Automata Product We now introduce constructions to deal with the union and intersection of automata on linear orderings originally introduced in [20, 38]. These constructions relies on automata product, which consists in building an automaton capable of simulating the synchronized behavior of two automata on identical input words. For the union, the resulting automaton must enforce that any accepting run it simulates possesses a corresponding counterpart in at least one of the factor automata, respecting the (successor and limit) transitions and the initial and accepting conditions. For the intersection, the resulting automaton must enforce that any accepting run it simulates corresponds to accepting runs in both factor automata.

Let $\mathcal{A}_1 = (Q_1, \Sigma_1, I_1, F_1, \Delta_1^s, \Delta_1^\ell)$ and $\mathcal{A}_2 = (Q_2, \Sigma_2, I_2, F_2, \Delta_2^s, \Delta_2^\ell)$ be two automata on linear orderings. We assume w.l.o.g. that the sets Q_1 and Q_2 are disjoint, i.e., $Q_1 \cap Q_2 = \emptyset$. An automaton $\mathcal{A}_1 \cup \mathcal{A}_2 = (Q_\cup, \Sigma_\cup, I_\cup, F_\cup, \Delta_\cup^s, \Delta_\cup^\ell)$ recognizing the language $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ can be defined by:

- The set of states $Q_\cup = (Q_1 \cup Q_2)$.
- The alphabet $\Sigma_\cup = \Sigma_1 \cup \Sigma_2$.
- The set of initial states $I_\cup = I_1 \cup I_2$.
- The set of accepting states $F_\cup = F_1 \cup F_2$.
- The set of successor transitions $\Delta_\cup^s = \Delta_1^s \cup \Delta_2^s$.
- The set of limit transitions $\Delta_\cup^\ell = \Delta_1^\ell \cup \Delta_2^\ell$.

And an automaton $\mathcal{A}_1 \cap \mathcal{A}_2 = (Q_\cap, I_\cap, F_\cap, \Delta_\cap^s, \Delta_\cap^\ell)$ recognizing the language $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ can be defined by:

- The set of states $Q_\cap = Q_1 \times Q_2$.
- The alphabet $\Sigma_\cap = \Sigma_1 \cap \Sigma_2$.
- The set of initial states $I_\cap = I_1 \times I_2$.
- The set of accepting states $F_\cap = F_1 \times F_2$.

- The set of successor transitions

$$\Delta_{\cap}^s = \{((q_1, q_2), \alpha, (q'_1, q'_2)) \mid (q_1, \alpha, q'_1) \in \Delta_1^s \wedge (q_2, \alpha, q'_2) \in \Delta_2^s\}.$$

- The set of limit transitions

$$\begin{aligned} \Delta_{\cap}^{\ell} = & \{P \rightarrow (q_1, q_2) \mid \{p_1 \mid (p_1, p_2) \in P\} \rightarrow q_1 \in \Delta_1^{\ell} \\ & \wedge \{p_2 \mid (p_1, p_2) \in P\} \rightarrow q_2 \in \Delta_2^{\ell}\} \\ & \cup \{(q_1, q_2) \rightarrow P \mid q_1 \rightarrow \{p_1 \mid (p_1, p_2) \in P\} \in \Delta_1^{\ell} \\ & \wedge q_2 \rightarrow \{p_2 \mid (p_1, p_2) \in P\} \in \Delta_2^{\ell}\}. \end{aligned}$$

Chapter 2

Drawing the Frontier of Decidability

In this chapter, we present an overview of the state of the art regarding the decidability of various theories gravitating around the theory of order. We particularly focus on fragments of second-order theories that involve some arithmetic, over classical domains such as $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{R}$, or more general classes of domains. The goal is to draw an accurate picture of the frontier of decidability when arithmetic and second-order logic are involved over various linearly ordered domains.

We introduced in Section 1.1.2 the fundamentals of first-order and second-order logics, and in particular we defined the notions of decidability and undecidability of a given theory. In this chapter, unless stated otherwise explicitly, we interpret decidability w.r.t. satisfiability, i.e., a theory is decidable if there exists an algorithm that, for any given formula φ of this theory, decides whether φ is satisfiable or not. Notice however that it is often the case that algorithms witnessing decidability of a theory are purely theoretical, and their complexity is too large to be of practical use. They may also be highly non-deterministic. On the other hand, a proof of undecidability consists in reducing an already known to be undecidable problem to the satisfiability problem for this theory. Such a proof usually relies on either a reduction from the *halting problem* of a *Turing machine* (we refer to, e.g., [107] for an introduction to the theory of computation in general, and Turing machines in particular), or a reduction from the satisfiability of another theory already proven undecidable.

First-Order with Arithmetic Let us start by examining the frontier of decidability in the context of strict first-order logic with arithmetic, i.e., second-order variables (even unquantified) are forbidden in formulas, but some arithmetic functions are allowed. We already mentioned in Example 2 the fact that Presburger arithmetic $\text{FO}(\mathbb{N}, <, 0, 1, +)$ is decidable [85], while Peano arithmetic $\text{FO}(\mathbb{N}, <, 0, 1, +, \times)$ is not [117, 33]. Skolem arithmetic $\text{FO}(\mathbb{N}, <, \times)$, another

related theory, is decidable [81]. It is worth mentioning that replacing natural numbers by integers in the three aforementioned theories, i.e., substituting $\mathbb{N}$ by $\mathbb{Z}$, has no impact on their decidability.

Interestingly enough, the counterpart $\text{FO}(\mathbb{R}, <, +, \times)$ of Peano arithmetic over the reals is decidable. More generally, the decidability of a theory of the form $\text{FO}(\mathbb{D}, <, +, \times)$ holds any for *real closed field* $\mathbb{D}$ [112]. Various extensions of $\text{FO}(\mathbb{R}, <, +, \times)$ have been considered. For instance, extending this fragment with the sine function yields undecidability (a consequence of Richardson's theorem [96, 31, 119]), while extending it with the exponential function preserves decidability [74, 120], provided Schanuel's conjecture is assumed.

The decidability of theories that combine the reals and the integers have also been investigated. For instance, the theory $\text{FO}(\mathbb{R}, <, +, \mathbb{Z})$, i.e., the first-order theory of order over the reals with addition, and a predicate for integer numbers, is decidable [110]. The theory $\text{FO}(\mathbb{R}, <, +, \times, \mathbb{Z})$ is undecidable since it is strictly more expressive than Peano arithmetic. The study of which restricted forms of multiplication can be incorporated into $\text{FO}(\mathbb{R}, <, +, \mathbb{Z})$ while preserving decidability is investigated in [63].

Introducing Second-Order Let us now change our perspective, and look at the frontier of decidability when second-order logic comes into play, without necessarily having arithmetic explicitly involved. A first central result is the decidability of the monadic first-order theory of order¹ (interpreted over the class of all linear orderings) $\text{MFO}(<)$, first established in [44]. Another proof of this result is presented in [73] which provides a useful insight at the expressive power of this theory.² Namely, if a formula of this theory is satisfiable, then it admits a simple model which results from a bounded application of some specific operators (loosely corresponding to concatenation, reverse and normal infinite repetitions, and the shuffle operator introduced in Section 1.3.1) applied to elementary components. A similar result naturally follows for the more restrictive cases of all ordinals, and all scattered linear orderings, noteworthily independent of the countability of the orderings. The decidability of the monadic first-order theory of order of the reals $\text{MFO}(\mathbb{R}, <)$ was proved in [25]. Its counterparts over the classical domains $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$ are also decidable, but this is a trivial consequence of stronger results we present thereafter. The combination of $\text{MFO}(<)$ with arithmetic can however quickly result in undecidable theories. For instance, the universal fragment³ of Presburger arithmetic augmented with a single unary uninterpreted predicate is undecidable [42, 62, 66].

Without much surprise, allowing quantification over second-order variables renders many of their MFO counterparts undecidable. The theory $\text{MSO}(\mathbb{N}, <, +)$ and $\text{MSO}(\mathbb{N}, <, x \mapsto 2x)$ were both shown to be undecidable in respectively [116] and [99]. Büchi [22] showed that the monadic second-order theory of

¹This theory is sometimes also referred to as *FOMLO* in the literature.

²The proof actually discards monadic predicates altogether and only covers the first-order theory of order, but the authors briefly state that the principles of the proof can be adapted to cover the monadic case.

³I.e., every formula is of the form $\forall x_1 \dots \forall x_n. \varphi$ where φ is a quantifier-free formula.

one successor over the naturals $\text{MSO}(\mathbb{N}, +1)$, also known as S1S, is decidable. His proof relies on the equivalence between this fragment and the eponymous Büchi automata. The principles of automata-based decision procedures will be further discussed in Chapter 4. The S1S theory considered by Büchi is actually equivalent to $\text{MSO}(\mathbb{N}, <)$, since the order relation over the natural numbers can be easily reconstructed using the successor operator $+1$, and reciprocally:

$$x < y \equiv \neg(x = y) \wedge \forall P, z. ((P(z) \Rightarrow P(z + 1)) \wedge P(x)) \Rightarrow P(y)$$

$$x = y + 1 \equiv x < y \wedge \neg(\exists z. x < z \wedge z < y)$$

In that context, the expressive power yielded by either considering the successor operator, or the order over the natural numbers are equivalent. Notice however that second-order quantification is necessary to reconstruct the order relation from the successor function, while the other direction only requires first-order quantification. The question of which (interpreted) predicates over $\mathbb{N}$ can be added to $\text{MSO}(\mathbb{N}, <)$ while preserving decidability has been investigated in, e.g., [46, 7, 6]. The decidability of $\text{MSO}(\mathbb{Z}, <)$ is a straightforward consequence of the decidability of $\text{MSO}(\mathbb{N}, <)$.

As a consequence of the decidability of the monadic second-order theory over infinite trees⁴ [87], the monadic second-order theory of order over countable linear orderings is also decidable. Intuitively, it is easy to translate a formula over linear orderings into a formula over infinite binary trees, and to describe a subset of an infinite binary tree order-isomorphic to the rational numbers. This notably implies the decidability of $\text{MSO}(\mathbb{Q}, <)$. The decidability of many other theories are a consequence of the results in [87], see in particular [88] for a survey. In [23], Büchi establishes the decidability of the monadic second-order theory of order over the class of countable ordinals, i.e., ordinals strictly less than ω_1 . He also proved that any satisfiable formula over countable ordinals admits a model defined over an ordinal strictly less than ω^ω . The extension of MSO over ordinals where it is possible to express that a subset of the domain (i.e., the interpretation of a second-order variable) is of order type ω^ω is actually undecidable [37]. Büchi later extended his decidability result to all ordinals strictly less than ω_2 in [24], where ω_2 is the initial ordinal associated with the cardinality $\aleph_2$. The case of ω_2 included has been shown to be independent from ZFC [57].

On the other hand, the monadic second-order theory of order over the reals, or $\text{MSO}(\mathbb{R}, <)$, was shown to be undecidable in [105]. The original proof by Shelah [105] relied on the continuum hypothesis, but this assumption can be lifted [59]. Its universal and existential fragments, i.e., when only one second-order quantifier is allowed to appear in a formula, are however decidable [25]. Various restrictions on the second-order quantification preserving decidability have been considered, in the sense that the second-order variables must be interpreted as subsets of $\mathbb{R}$ satisfying certain properties. The decidability of the

⁴The theory over binary trees is also called S2S, following the denomination of S1S, but with two successors.

weak monadic second-order theory of order over all linear orderings was proved in [72], where ‘weak’ means that second-order quantifiers can only range over finite subsets of the domain. Another consequence of [87] is the decidability of the monadic second-order theory of order over $\mathbb{R}$ where the second-order variables are restricted to countable unions of closed sets (also called F_σ sets). More recently, this result has been extended to the case where second-order variables are restricted to be any Borel sets [77], that is, certain combinations of open and closed sets. Related questions about the definability of certain sets of integers or rationals with the real order as the background have been investigated in [58, 30].

Linear Temporal Logic In parallel to the first and second-order theories, other forms of logics have been considered to express properties over linearly ordered domains. In particular, a lot of interest has been focused on linear temporal logic (LTL), where formulas relate to the past, the present, and the future of events over a linear time.⁵ Temporal logics, initially introduced in a philosophical setting [86] and then considered for computer science by Pnueli [84], provide a convenient framework to express classical verification properties for computer programs, such as safety or fairness. LTL relies on specific operators called modalities that allow to describe behaviors over various time intervals. Among the most common modalities are the unary *Always* (or *Globally*, noted $\Box$) and *Eventually* (or *Finally*, noted $\Diamond$), which respectively express that a specific property holds at every moment in time, or that there exists a moment in the future at which the property will hold, and the binary *Until* and *Since*, which respectively express that a specific property will hold at every moment in time until another specific property becomes true for the first time in the future, or that since a specific property became true, another property hold at every moment in time up to now. One could, for instance, express that it is always the case that after a certain event P occurs, an event R will eventually occur in the future. Interpreting, e.g., P as a client requesting access to a shared resource, and R as the resource being accessed by the client, gives an illustration of how to express a fairness property of a concurrent system in this framework.

We will not go much deeper into the specifics of temporal logics, and refer to, e.g., [76] for a general introduction to LTL. Let us however discuss the connections between LTL and the monadic first-order logic of order. Namely, a major result about LTL is Kamp’s Theorem [67], which states the equivalence between the monadic first-order theories of order and LTL with modalities *Until* and *Since* over any Dedekind complete domain, such as $\mathbb{N}$, $\mathbb{R}$, or the class of all ordinals for instance. This equivalence can be extended to cover all linear orderings [53], i.e., also cover incomplete domains such as $\mathbb{Q}$, by considering two additional modalities *Until_{Stavi}* and *Since_{Stavi}*. Alternate shorter proofs of Kamp’s theorem have been developed in [52, 91].

⁵Not all temporal logics consider a linear time. Some, such as CTL [47] are branching-time logics, where interpretations may involve multiple possible futures at once.

Proof Techniques for Decidability We just review numerous previous results on the decidability and undecidability of theories that involve arithmetic combined with second-order logic. We now give some insights into two families of techniques on which most decidability results presented beforehand rely.

The first technique relies on automata theory to design a decision procedure. Let us briefly recall the fundamentals of automata-based decision procedures. A more in-depth presentation of this approach will be given in Chapter 4. The principle is, for a given theory, to define a way to encode any possible interpretation in this theory into a word that can be read by an automaton. E.g., using infinite words (or ω -words) for theories interpreted over the domain $\mathbb{N}$ as in [22], or labeled trees for theories interpreted over classes of trees as in [87]. Then, for any formula in the considered theory, one builds a corresponding automaton that recognizes exactly the language of the encodings of the models of this formula. Finally, one decides the satisfiability of the input formula by checking the non-emptiness of the language recognized by the obtained automaton.

The second approach, usually described as ‘model theoretic’, is the *composition method*. The idea behind this method is to view interpretations as compositions of simpler structures. The composition operators can be multiple (such as concatenation, Cartesian product, ...) and depend on the type of theory involved. This method actually extends beyond linear orderings, and can be used for, e.g., theories over graphs [122].

The composition method relies on a notion of equivalence between interpretations initially introduced by Ehrenfeucht [45] and Fraïssé [51]. More precisely, two interpretations⁶ are considered k -equivalent over a certain theory, if they cannot be differentiated by a formula in that theory with a quantifier depth less than or equal to k , with $k \in \mathbb{N}$. For every $k \in \mathbb{N}$, this notion of equivalence yields a finite set of equivalence classes. Each of which can be represented by a characterizing formula, called a *Hintikka formula* [64].

We now briefly recall the history of the introduction and use of this technique. Feferman and Vaught [49], inspired by earlier work of Mostowski [81], introduced the composition method to prove the decidability of some purely first-order theories. This technique was then extended to obtain the decidability of monadic second-order theories by Laüchli [72] and Shelah [105]. We will not try to list all decidability results obtained via the composition method, but we refer to [114, 122, 75] for thorough presentations of the technique, illustrations of its uses, and to a survey by Gurevich [59], another important contributor of the development of this technique.

The alternate proof of the decidability of $\text{MFO}(\mathbb{R}, <)$ that we will present in Section 3.3 relies on the composition method, and the required formal definitions will be recalled there.

⁶Here the notion of interpretation slightly differs from its definition in Section 1.1. Intuitively, it is not associated to a unique formula anymore, but to any formula involving the right number of first-order and second-order free variables. More will be said about this in Section 3.3.

A Few Words on Complexity For now, we only talked about decidability and undecidability of theories. But the decidability of a theory directly gives rise to another question: In which complexity class does the decidability problem associated with this theory belong? (We refer the reader to, e.g., [83] for a general introduction to complexity theory.) Without going into too many details, let us state some central complexity results, relevant to our main goal to design effective decision procedures for various theories of order.

The decision procedure introduced by Büchi to decide S1S [22] has a non-elementary worst-case complexity, that is, a tower of exponential proportional in height to the length of the input formula. The reason is that the elimination of universal quantifiers relies on complementation of Büchi automata, which is a costly operation [102, 118] that may incur an exponential blowup in the number of states of resulting automata. As mentioned previously, this approach will be further discussed in Chapter 4. The complexity of related decision problems linking logic and theory of automata have been investigated in [80, 111].

The complexity of deciding LTL is investigated in [93, 94, 95], where it is established in particular that this problem is PSPACE-complete for LTL with only the modality *Until* over all linear orderings, as well as for LTL with *Since* and *Until* over $\mathbb{R}$. A simpler proof — noteworthy combining elements of the composition method with automata theory — of those results appears in [90]. Note that this does not lead to a PSPACE decision procedure for the monadic first-order theory of order over the reals $\text{MFO}(\mathbb{R}, <)$, since translating a formula of this logic into LTL may incur a non-elementary blowup [91].

It is worth observing that a high worst-case complexity does not automatically translates into the impossibility of designing practical and effective decision procedures. Let us illustrate this with a few examples of effective implementations of decision procedures. The tool MONA [69] decides a simpler fragment of S1S, denoted WS1S — for *Weak Second-order theory of 1 Successor* — where second-order variables can only range over finite sets. The tool QEPCAD [19] deals with the theory of real closed fields.

In a related field, the NP-completeness of SAT — i.e., the *Boolean satisfiability problem* — does not prevent SAT solvers to meet great practical success [54]. And in formal verification, it is standard to develop sound and efficient but incomplete tools [40, 9]. In that sense, even undecidability does not necessarily constitutes an absolute obstacle to the design of effective procedures.

Chapter 3

Decidable and Undecidable Theories: Contributions

In this chapter, we present three contributions that participate in the refinement of the frontier of decidability. These results originally appeared in [11] and [13]. We start by proving the decidability of the monadic first-order theory of order over the reals, extended with an interpreted predicate describing the integers, and *difference constraints*¹ restricted to only involve *integer-guarded* variables. We then show that the theory $M_1FO(\mathbb{R}, <, +1)$, i.e., the monadic first-order theory of order over the reals, with a $+1$ function, and a single unary uninterpreted predicate is undecidable. Finally, we provide an alternate proof of the decidability of the monadic theory of order interpreted over the reals $MFO(\mathbb{R}, <)$.

3.1 The Case of the Theory of Order over the Reals with Difference Logic over the Integers

In the satisfiability modulo theory (SMT) world, *difference logic* is a quite popular fragment of linear arithmetic which is less expressive than Presburger arithmetic. In difference logic, the arithmetic atoms in formulas can only take the form $x - y \bowtie c$, where x and y are first-order variables, $c \in \mathbb{Z}$ is an integer constant, and $\bowtie \in \{<, \leq, =, \geq, >\}$. These atoms are simply syntactic sugar that can easily be reconstructed if one has access to a $+1$ function, e.g.,

$$x - y = 3 \equiv \exists z_1, z_2. z_1 = y + 1 \wedge z_2 = z_1 + 1 \wedge x = z_2 + 1$$

Difference logic does not admit atoms of the form $x - y = z$, where x , y , and z are variables. Such atoms are however allowed in theories that have the full

¹A formalism that allows to express properties on the distance between two variables. The formal definitions is provided thereafter, in Section 3.1

power of addition. This is the key distinction between the binary function ‘+’ and the unary function ‘+1’.

In this section, we prove the decidability of the monadic first-order theory of order over the reals, extended with an interpreted predicate describing the integers, and difference constraints restricted to only involve *integer-guarded* variables. We denote this theory by $\text{MFO}(\mathbb{R}, <, \mathbb{Z}, +1_{[\mathbb{Z}]})$, where $+1_{[\mathbb{Z}]}$ denotes the interpreted function from $\mathbb{Z}$ to $\mathbb{Z}$ defined as the usual successor function over the integers. More precisely, this means that in addition to the usual atoms allowed in the monadic first-order theory of order, atoms of the form $x \in \mathbb{Z}$ are allowed (where x is a first-order variable), and that the only difference constraints allowed are of the form $x \in \mathbb{Z} \wedge y \in \mathbb{Z} \wedge x - y \bowtie c$ or $(x \in \mathbb{Z} \wedge y \in \mathbb{Z}) \Rightarrow x - y \bowtie c$, where x and y are first-order variables, $c \in \mathbb{Z}$ is an integer constant, and $\bowtie \in \{<, \leq, =, \geq, >\}$.

Example 8. *The following formula in the theory $\text{MFO}(\mathbb{R}, <, \mathbb{Z}, +1_{[\mathbb{Z}]})$ describes a predicate that is either true on all even numbers and false on all odd numbers, or the opposite, as well as true on all non-integer numbers.*

$$\begin{aligned} & \forall x, y. (x \in \mathbb{Z} \wedge y \in \mathbb{Z} \wedge y - x = 2) \Rightarrow (P(x) \Leftrightarrow P(y)) \\ & \wedge \exists x, y. x \in \mathbb{Z} \wedge y \in \mathbb{Z} \wedge P(x) \wedge \neg P(y) \wedge (\forall z. \neg(z \in \mathbb{Z}) \Rightarrow P(z)) \end{aligned}$$

The remainder of this section reproduces² Section 3 of the following article [11]:

Bernard Boigelot, Pascal Fontaine, **Baptiste Vergain**. Decidability of difference logic over the reals with uninterpreted unary predicates. In *International Conference on Automated Deduction*, pages 542-559, Springer, 2023.

The proof of decidability consists in a reduction to the monadic first-order theory of order over the reals, or $\text{MFO}(\mathbb{R}, <)$, known to be decidable [25]. More precisely, we describe a simple procedure to translate any formula in $\text{MFO}(\mathbb{R}, <, \mathbb{Z}, +1_{[\mathbb{Z}]})$ into an equisatisfiable formula in $\text{MFO}(\mathbb{R}, <)$.

Theorem 3.1.1. *The theory $\text{MFO}(\mathbb{R}, <, \mathbb{Z}, +1_{[\mathbb{Z}]})$ is decidable.*

3.1.1 Recognizing Integer Values

We first show how to define in $\text{MFO}(\mathbb{R}, <)$ a predicate P_{int} over $\mathbb{R}$ that is order-isomorphic to $\mathbb{Z}$. Integer guards (i.e., atoms of the form $x \in \mathbb{Z}$) will later be translated using P_{int} .

We axiomatize P_{int} in $\text{MFO}(\mathbb{R}, <)$ as follows:

- Every element of P_{int} is isolated:

$$\forall x \exists y, z. y < x < z \wedge \forall t. (y < t < z \wedge P_{\text{int}}(t)) \Rightarrow t = x.$$

²Some notations have been adapted to match the nomenclature introduced in Chapter 1.

- Every point in $\mathbb{R}$ has a unique successor in P_{int} :
 $\forall x \exists y. x < y \wedge P_{int}(y) \wedge \forall t. x < t < y \Rightarrow \neg P_{int}(t).$
- Similarly, every point in $\mathbb{R}$ has a unique predecessor in P_{int} :
 $\forall x \exists y. y < x \wedge P_{int}(y) \wedge \forall t. y < t < x \Rightarrow \neg P_{int}(t).$

Notice that the successor and predecessor axioms ensure that P_{int} is not the empty predicate. Furthermore, it is worth mentioning that defining the successor (resp. predecessor) axiom by stating that every point in P_{int} has a successor (resp. predecessor) in P_{int} would not be correct, as P_{int} could then also describe sets that are not order-isomorphic to $\mathbb{Z}$, e.g., sets of order type $-\omega \cdot \omega \cdot -\omega \cdot \omega$.

The set of all integers is a model for P_{int} , therefore the above axiomatization is consistent. The set of elements satisfying P_{int} is necessarily infinite and does not admit a maximal or a minimal element. This is a direct consequence of the successor and predecessor axioms. More interestingly, this set is also necessarily countable. Indeed, since each point is isolated, there exists an application that maps the elements satisfying P_{int} to disjoint open intervals. Any set of disjoint intervals in $\mathbb{R}$ with non-zero lengths is necessarily countable [106], since each of them contains a rational value that does not belong to the others.

It is now possible to define a successor relation on the real numbers satisfying P_{int} with the formula

$$Succ(x, y) = P_{int}(x) \wedge P_{int}(y) \wedge y < x \wedge \forall z. y < z < x \Rightarrow \neg P_{int}(z),$$

which expresses that x is the successor of y , or equivalently, y is the predecessor of x . The axiomatization of P_{int} is, in fact, precise enough to have the following lemma.

Lemma 3.1.1. *For any model M of P_{int} , the set $M(P_{int})$ is order-isomorphic to $\mathbb{Z}$.*

For convenience in the proof, we define 0_{int} as an arbitrary existentially quantified value that belongs to the set described by P_{int} .

Proof. Given a model M of the axiomatization of P_{int} , we need to define a bijection between the set $M(P_{int})$ and $\mathbb{Z}$ that preserves order.

Let us define an application f from $M(P_{int})$ to $\mathbb{Z}$. We set $f(0_{int}) = 0$, and then define recursively:

- $f(y) = f(x) + 1$ for each $x, y \in M(P_{int})$ such that $y > 0_{int}$ and $Succ(y, x)$,
- $f(y) = f(x) - 1$ for each $x, y \in M(P_{int})$ such that $y < 0_{int}$ and $Succ(x, y)$.

Thanks to the fact that every element of $M(P_{int})$ has a unique predecessor and a unique successor, it follows that f ranges over the whole set $\mathbb{Z}$, proving that f is surjective. Since it is clear that f preserves order, it follows that f is strictly increasing, and therefore injective. It remains to show that f is well defined for every element in $M(P_{int})$.

If there exists some element $y \in M(P_{int})$ for which f is not defined, it means that f is not well-defined, in the sense that there exists either an element $y > 0_{int}$ such that the interval $[0_{int}, y]$ contains an infinite number of elements satisfying P_{int} , or there exists an element $y < 0_{int}$ such that the interval $[y, 0_{int}]$ contains an infinite number of elements satisfying P_{int} . Since both cases are symmetric, we only address the former. There must exist a strictly increasing infinite series of elements in $M(P_{int})$ bounded by y . Let us consider its limit $z \in \mathbb{R}$. Because there must exist an element of $M(P_{int})$ smaller than z and arbitrarily close to z , it follows that z cannot have a predecessor, which contradicts an axiom. Therefore f is well-defined, and every element of $M(P_{int})$ is associated to an integer number. The application f is therefore a bijection. $\square$

3.1.2 Translating Formulas

We are now ready to describe the satisfiability-preserving translation of formulas from $\text{MFO}(\mathbb{R}, <, \mathbb{Z}, +1_{[\mathbb{Z}]})$ to $\text{MFO}(\mathbb{R}, <)$. Consider a $\text{MFO}(\mathbb{R}, <, \mathbb{Z}, +1_{[\mathbb{Z}]})$ formula φ . Without loss of generality, we assume that the symbol P_{int} does not appear in φ . The translation of φ is defined as

$$AXIOMS_{int}(P_{int}) \wedge \llbracket \varphi \rrbracket$$

where $AXIOMS_{int}(P_{int})$ is the conjunction of the axioms of P_{int} , and $\llbracket \cdot \rrbracket$ is a translation operator. This translation operator $\llbracket \cdot \rrbracket$ distributes over all Boolean operators and quantifiers, and corresponds to the identity transformation for most considered atoms, except in the following cases:

- $\llbracket x \in \mathbb{Z} \rrbracket = P_{int}(x)$;
- $\llbracket x - y \bowtie c \rrbracket = \exists z_0, \dots, z_c. (y = z_0) \wedge (x \bowtie z_c) \wedge \bigwedge_{0 \leq i < c} \text{Succ}(z_{i+1}, z_i)$,
for $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. We assume that $z_0, \dots, z_c$ are fresh variables w.r.t. x and y .

Example 9. The translation $\llbracket x - y \leq 2 \rrbracket$ is the $\text{MFO}(\mathbb{R}, <)$ formula

$$\exists z_0, z_1, z_2. y = z_0 \wedge \text{Succ}(z_1, z_0) \wedge \text{Succ}(z_2, z_1) \wedge x \leq z_2.$$

Notice that we only deal with the case $c \in \mathbb{N}$ since every atom of the form $x - y \bowtie c$ with $c \in \mathbb{Z} \setminus \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$ can be rewritten as $y - x \bowtie' -c$ with the following correspondences: $(\bowtie, \bowtie') \in \{ (=, =), (<, >), (>, <), (\geq, \leq), (\leq, \geq) \}$.

3.1.3 Establishing Equisatisfiability

It remains to establish that any formula ψ obtained via the translation is equisatisfiable to the original formula φ , i.e., we need to prove that if φ admits a model, then ψ also admits one, and reciprocally.

Lemma 3.1.2. *Given a $\text{MFO}(\mathbb{R}, <, \mathbb{Z}, +1_{[\mathbb{Z}]})$ formula φ , consider its translation $\psi = AXIOMS_{int}(P_{int}) \wedge \llbracket \varphi \rrbracket$ in $\text{MFO}(\mathbb{R}, <)$. The formulas φ and ψ are equisatisfiable.*

Proof. If φ is satisfiable, then let M be one of its models. Since ψ shares the same free variables and predicates than φ with the only addition of P_{int} , we can directly construct a model M' of ψ that is similar to M for the shared variables and predicates, and that interprets P_{int} so that $P_{int}(x)$ holds whenever $x \in \mathbb{Z}$. This is always possible since the only constraints on P_{int} generated by the construction of ψ are the axioms stated above.

If ψ is satisfiable, then there exists a model M of ψ . Let us construct a model M' of φ . Let $0_{int} \in \mathbb{R}$ be an arbitrary element of $M(P_{int})$. We define an automorphism g of $\mathbb{R}$ — that is, an order-preserving bijection from $\mathbb{R}$ to $\mathbb{R}$ — such that $g(0_{int}) = 0$, and recursively $g(y) = g(x) + 1$ for $x, y \in M(P_{int})$, $y > 0_{int}$ and $Succ(y, x)$, and $g(y) = g(x) - 1$ for $x, y \in M(P_{int})$, $y < 0_{int}$ and $Succ(x, y)$. The automorphism g maps each open interval between the k -th and $(k+1)$ -th successors (resp. predecessors) of 0_{int} in $M(P_{int})$, onto the open interval $(k, k+1)$ (resp. $-(k+1), -k$) while preserving order.

M' is defined by $M'(x) = g(M(x))$ for each free variable x of the formula φ , and $M'(P) = \{g(x) \mid x \in M(P)\}$ for each uninterpreted predicate P of φ . No unary predicate atom can be violated by M' by definition. Furthermore, no order constraint can be violated by M' either since g preserves order. Regarding the difference logic constraints, the intermediate variables z_i introduced in the translation are necessarily mapped to values in $M(P_{int})$ since the $Succ$ relation enforces this property. Hence for each such variable, we have $g(M(z_i)) \in \mathbb{Z}$. Intuitively, this ensures that in M' the difference between the values taken by the integer variables is consistent with the difference logic constraints. It follows that M' is a model of φ . $\square$

3.2 The Case of Difference Logic over the Reals with One Predicate Symbol

This section is dedicated to the proof of undecidability of $M_1FO(\mathbb{R}, <, +1)$, i.e., the monadic first-order theory of order over the reals, with a $+1$ function, and a single unary uninterpreted predicate. We would like to mention that the undecidability of $MFO(\mathbb{R}, <, +1)$ seems to be considered as an accepted fact (as mentioned in [65], Section 3.1, for instance), but we did not find an explicit proof in the literature, nor any mention of the case where only a single unary predicate is allowed.

The remainder of this section reproduces Section 4 of the following paper [11]:

Bernard Boigelot, Pascal Fontaine, **Baptiste Vergain**. Decidability of difference logic over the reals with uninterpreted unary predicates. In *International Conference on Automated Deduction*, pages 542-559, Springer, 2023.

Theorem 3.2.1. *The theory $M_1FO(\mathbb{R}, <, +1)$ is undecidable.*

In the following development, we consider w.l.o.g. Turing machines defined over an alphabet with only two symbols and no explicit blank symbol [104]. This choice leads to a simpler proof.

3.2.1 Definitions

The proof is by reduction from the halting problem for a Turing machine with a single bi-infinite tape, starting from a blank tape (i.e., a tape filled with the symbol 0). Consider a Turing machine $\mathcal{M} = (Q, \Sigma, q_I, q_F, \Delta)$, where

- Q is a finite nonempty set of states,
- Σ is the alphabet $\{0, 1\}$,
- $q_I \in Q$ is the initial state,
- $q_F \in Q$ is the halting state,
- $\Delta \subseteq ((Q \setminus \{q_F\}) \times \Sigma \times Q \times \Sigma \times \{L, R\})$ is the transition relation, assumed to be total over its first two components, i.e., for any pair $(q, \alpha) \in (Q \setminus \{q_F\}) \times \Sigma$, there exists a tuple $(q, \alpha, q', \alpha', \lambda) \in \Delta$.

A *configuration* C of such a Turing machine is a triplet containing the current state q , the content of the tape $t \in \{0, 1\}^{\mathbb{Z}}$ and the position of the head $h \in \mathbb{Z}$. Since the machine starts from a blank tape, the initial configuration is $C_0 = (q_I, 0^{\mathbb{Z}}, 0)$.

A *run* ρ of length $n \in \mathbb{N}$ (resp. $n = +\infty$) of such a Turing machine is a finite (resp. infinite) sequence of configurations $(C_i)_{i \in [0, n]}$ (resp. $(C_i)_{i \in \mathbb{N}}$), such that for any two consecutive configurations $C_i = (q_i, t_i, h_i)$ and $C_{i+1} = (q_{i+1}, t_{i+1}, h_{i+1})$ there exists a transition $(q, \alpha, q', \alpha', \lambda) \in \Delta$ such that:

- $q = q_i$ and $q' = q_{i+1}$,
- $t_i[h_i] = \alpha$, i.e., the tape cell at position h_i contains the symbol α ,
- $t_{i+1}[h_i] = \alpha'$,
- $t_{i+1}[k] = t_i[k]$, for every $k \in \mathbb{Z}$, $k \neq h_i$,
- $h_{i+1} = h_i + 1$ if $\lambda = R$, and $h_{i+1} = h_i - 1$ if $\lambda = L$.

A *halting run* is a finite run such that the state of its last configuration is the halting state q_F .

3.2.2 Encoding Runs

Our goal is to encode a run of a Turing machine (as described before), i.e., encode the state, the tape content, and the position of the head for each configuration of such a run. Starting from the initial configuration, we must also ensure the coherence of the run w.r.t. the Turing machine transition relation,

by connecting every two consecutive configurations. Our idea is to define an infinite sequence of intervals on the real line, such that each interval contains the encoding of its corresponding configuration (i.e., the first interval will contain the first configuration of the run, and so on). Difference constraints can then be used to connect consecutive configurations.

Let $N = \lceil \log_2(|Q|) \rceil$. Each state $q \in Q$ of $\mathcal{M}$ can therefore be uniquely encoded with N Boolean values $b_1^q, \dots, b_N^q$. We want to encode consecutive configurations of the Turing machine using a single predicate P over $\mathbb{R}$. In order to do so, we first need to describe a subset of $\mathbb{R}$ that will act as a grid supporting the encoding of the state, the tape content, and the head position of the current configuration.

The grid we consider is a subset of $\mathbb{R}$, of order type $(N + \zeta + 1 + \zeta)^\omega$. An ordering of order type $N + \zeta + 1 + \zeta$ within the interval $[0, 3]$ is depicted in Figure 3.1. Each dot corresponds to a natural number and each vertical line corresponds to an element of the linear ordering. The first N points will support the encoding of a state. The first subordering that is order-isomorphic to $\mathbb{Z}$ (i.e., of order type ζ) will be used to encode the position of the head, while the second one will support the encoding of the tape content. The whole grid is composed of an infinite repetition of the subordering $N + \zeta + 1 + \zeta$ (i.e., it is repeated on the intervals $[3k, 3k + 3]$ for all $k \in \mathbb{N}$), hence the ω exponent.

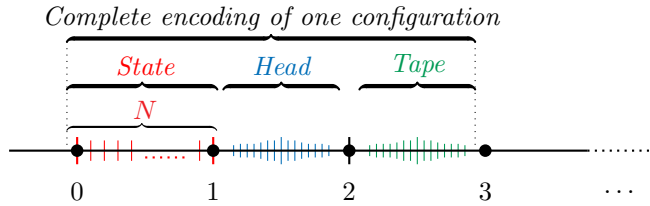


Figure 3.1: A visual representation of a linear ordering of order type $N + \zeta + 1 + \zeta$.

3.2.3 Defining the Support of the Encoding

Let us first define concretely the support of the encoding of the Turing machine configurations. The difficulty lies in describing the grid using a single predicate P , without meddling with the actual encoding of the configurations afterwards. Our solution is to characterize the points that belong to the grid by enforcing that such a point is surrounded by an open interval where P is uniformly *true* on the left, and by an open interval where P is uniformly *false* on the right, such as depicted in Figure 3.2. We do not specify yet how P behaves on x , as this is how the configurations will actually be encoded later.

Such a characterization is easy to express in $M_1FO(\mathbb{R}, <, +1)$:

$$\begin{aligned} \text{Support}(x) = & \exists y. y < x \wedge \forall z. y < z < x \Rightarrow P(z) \\ & \wedge \exists y. x < y \wedge \forall z. x < z < y \Rightarrow \neg P(z) \end{aligned}$$

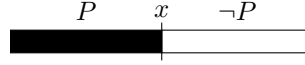


Figure 3.2: The real number x belongs to the grid, since it is surrounded by a *true* (black) open interval on the left, and a *false* (white) open interval on the right.

Let us now partially axiomatize the predicate P such that the set of *supporting* points constitutes a linear ordering of order type $(N + \zeta + 1 + \zeta)^\omega$:

- (a) Let $\mathbf{0}$ be a variable and $\mathbf{1}, \mathbf{2}$ and $\mathbf{3}$ be respectively the +1-successor of $\mathbf{0}$, $\mathbf{1}$ and $\mathbf{2}$:

$$Axiom_1 = (\mathbf{1} = \mathbf{0} + 1) \wedge (\mathbf{2} = \mathbf{1} + 1) \wedge (\mathbf{3} = \mathbf{2} + 1)$$

These free variables are implicitly existentially quantified in the final formula. Notice that the variable $\mathbf{0}$ can be interpreted as any real value, which only acts as a landmark for the beginning of the grid.

- (b) $\mathbf{0}, \mathbf{1}$ and $\mathbf{2}$ are *supporting* points:

$$Axiom_2 = Support(\mathbf{0}) \wedge Support(\mathbf{1}) \wedge Support(\mathbf{2})$$

- (c) P is uniformly true before $\mathbf{0}$, i.e., there are no supporting points before $\mathbf{0}$:
 $Axiom_3 = \forall x. x < \mathbf{0} \Rightarrow P(x)$

- (d) There are exactly $N - 2$ *supporting* points within the interval $(\mathbf{0}, \mathbf{1})$:

$$Axiom_4 = \exists x_1, x_2, \dots, x_N. x_1 = \mathbf{0} \wedge x_N = \mathbf{1} \\ \wedge \bigwedge_{1 \leq i < N} (\mathbf{0} \leq x_i < \mathbf{1} \wedge Succ_{Supp}(x_{i+1}, x_i))$$

where $Succ_{Supp}(x, y)$ is a formula that states that x is the first *supporting* real value that is strictly greater than y , i.e., x is the successor of y on the grid. It is defined as follows:

$$Succ_{Supp}(x, y) = y < x \wedge Support(x) \wedge Support(y) \\ \wedge \forall z. y < z < x \Rightarrow \neg Support(z)$$

We also define an analogous formula to express that x is the predecessor of y : $Pred_{Supp}(x, y) = Succ_{Supp}(y, x)$.

- (e) The set of *supporting* points within $(\mathbf{1}, \mathbf{2})$ is order-isomorphic to $\mathbb{Z}$. This is done similarly to the axiomatization of P_{int} (cf. Section 3.1.1). But because $\mathbf{1}$ (resp. $\mathbf{2}$) is a *supporting* point, there must exist a uniformly false (resp. true) interval of P at its right (resp. left) where no other *supporting* points can appear. All the *supporting* points will therefore be constrained to appear within a smaller interval (b_1, b_2) with $\mathbf{1} < b_1 < b_2 < \mathbf{2}$, as illus-

trated in Figure 3.3.

$$Axiom_5 = \exists b_1, b_2. \mathbf{1} < b_1 < b_2 < \mathbf{2} \quad (3.1)$$

$$\begin{aligned} \wedge (\forall x. (b_1 < x < b_2) \Rightarrow \exists y. x < y < b_2 \wedge Support(y) \\ \wedge \forall z. x < z < y \Rightarrow \neg Support(z)) \end{aligned} \quad (3.2)$$

$$\begin{aligned} \wedge (\forall x. (b_1 < x < b_2) \Rightarrow \exists y. b_1 < y < x \wedge Support(y) \\ \wedge \forall z. y < z < x \Rightarrow \neg Support(z)) \end{aligned} \quad (3.3)$$

$$\wedge (\forall x. (1 < x < 2 \wedge Support(x)) \Rightarrow b_1 < x < b_2) \quad (3.4)$$

This axiom can be broken down into these elementary pieces:

- (3.1) There exists an open interval (b_1, b_2) such that $\mathbf{1} < b_1 < b_2 < \mathbf{2}$.
- (3.2) Each real value in (b_1, b_2) has a *supporting* successor.
- (3.3) Each real value in (b_1, b_2) has a *supporting* predecessor.
- (3.4) There are no *supporting* points within $(\mathbf{1}, b_1)$, nor within $(b_2, \mathbf{2})$.
- (f) The pattern of *supporting* points within $(\mathbf{1}, \mathbf{2})$ is repeated onto the interval $(\mathbf{2}, \mathbf{3})$ with an exact offset of 1:

$$Axiom_6 = \forall x. \mathbf{1} < x < \mathbf{2} \Rightarrow (Support(x) \Leftrightarrow Support(x + 1))$$

- (g) The pattern of *supporting* points within $[0, \mathbf{3})$ is repeated onto every interval $[3k, 3k + 3)$ for $k \in \mathbb{N}$:

$$Axiom_7 = \forall x. x \geq \mathbf{0} \Rightarrow (Support(x) \Leftrightarrow Support(x + 3))$$

Notice that for $Axiom_7$, it is not enough that a similar pattern appears within each interval $[3k, 3k + 3)$: there must be an exact offset of 3 with the previous interval. This is mandatory to connect two consecutive configurations and ensure that they are coherent with the transition relation of the Turing machine, as defined later. The same goes for $Axiom_6$, where the exact offset of 1 will allow to connect the position of the head to the tape content within a single configuration.



Figure 3.3: The points of the grid surrounded by open *true* (black) and *false* (white) intervals within $(\mathbf{1}, \mathbf{2})$.

The formula $AXIOMS_{Supp} = \bigwedge_{1 \leq k \leq 7} Axiom_k$ axiomatizes the predicate P .

Lemma 3.2.1. *The formula $AXIOMS_{Supp}$ is consistent.*

The proof sketch below provides the key ideas to construct a model of $AXIOMS_{Supp}$.

Proof. Let us construct a subset S of $\mathbb{R}$ that is a model of $AXIOMS_{Supp}$. Firstly, we make every negative number belong to S , which ensures that there do not exist negative supporting points. The interval $[0, 1]$ is then cut into $2N - 2$ intervals of equal length, which alternate between being included in S , and being disjoint from S . This ensures the existence of exactly $N - 1$ supporting points within the interval $(-\infty, 1)$, 0 being the first; 1 will be considered later. These $N - 1$ supporting points are referred to as $s_1, s_2, \dots, s_{N-1}$ and are depicted in Figure 3.4. Recall that the supporting points are exactly those surrounded by an interval of S (i.e., black on the figure) on the left, and an interval disjoint from S (i.e., white) on the right.

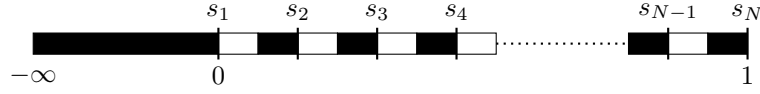


Figure 3.4: A model for the axiomatization of P over the interval $(-\infty, 1)$.

In order to make the real value 1 the N -th supporting point, it is enough to make an interval on its right disjoint from S , e.g., the interval $(1, 1 + \frac{1}{4})$. Symmetrically, we make the interval $(2 - \frac{1}{4}, 2)$ included in S , satisfying the left part of the requirement for the real value 2 to be a supporting point. We further characterize S such that the set of supporting points within the interval $(1 + \frac{1}{4}, 2 - \frac{1}{4})$ is order-isomorphic to $\mathbb{Z}$. This can be done by partitioning the open interval $(1 + \frac{1}{4}, 2 - \frac{1}{4})$ into a bi-infinite sequence of open intervals alternating between being included and disjoint from S , as depicted in Figure 3.5.

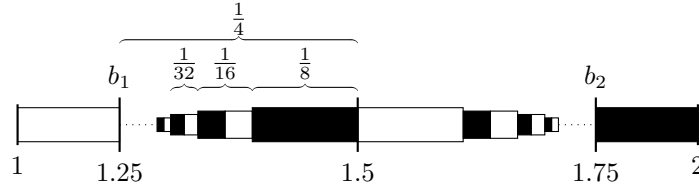


Figure 3.5: A model for the axiomatization of P over the interval $(1, 2)$.

The whole pattern described on the interval $(1, 2)$ can be directly transposed onto the interval $(2, 3)$ with an exact offset of $+1$. Similarly, the distribution of S over the interval $(0, 3)$ can be transposed onto every interval $(3k, 3k + 3)$ with an offset of $+3k$, for $k > 0$. The only real values for which we do not describe their relation with S are the points surrounded by an interval included in S on one side, and an interval disjoint from S on the other side. These points never conflict with the axiomatization $AXIOMS_{Supp}$ which only deals with non-empty open intervals.

By construction, S satisfies each axiom of the formula $AXIOMS_{Supp}$, and is therefore a model of this formula. $\square$

3.2.4 Encoding a Configuration of the Turing Machine

Now that the supporting grid has been properly defined, the actual encoding of a given configuration can be addressed. That is, the state, the tape content and the head position of the $(k+1)$ -th configuration of a run are encoded on the supporting points contained within the interval $[3k, 3k+3)$.

Encoding the State

Encoding the state of a given configuration is rather direct since we defined the grid to contain N consecutive supporting points within every interval $[3k, 3k+1]$ for $k \in \mathbb{N}$, that can support the encoding of a state. We only need to indicate that we start reading the encoding on a multiple of 3. However the logic $\text{FO}(\mathbb{R}, <, +1, P)$ does not allow to express periodicity constraints on variables.

Nevertheless, thanks to our axiomatization, 0 and every other positive multiple of 3 are the only points that simultaneously have no supporting predecessor, while admitting a supporting successor. These properties are expressible as follows:

$$\text{NoPred}_{\text{Supp}}(x) = \forall z. (z < x \wedge \text{Support}(z)) \Rightarrow \exists y. z < y < x \wedge \text{Support}(y)$$

$$\text{HasSucc}_{\text{Supp}}(x) = \exists z. x < z \wedge \text{Support}(z) \wedge \forall y. x < y < z \Rightarrow \neg \text{Support}(y)$$

For convenience, we introduce the formula *EncodingBegins* to characterize a real value x on which the encoding of a state starts:

$$\text{EncodingBegins}(x) = \text{Support}(x) \wedge \text{NoPred}_{\text{Supp}}(x) \wedge \text{HasSucc}_{\text{Supp}}(x)$$

Furthermore, the formula State_q expresses that a state $q \in Q$ is encoded on a given real number x and its $N-1$ supporting successors:

$$\begin{aligned} \text{State}_q(x) = & \text{EncodingBegins}(x) \wedge \exists y_1, \dots, y_N. x = y_1 \\ & \wedge \bigwedge_{1 \leq i < N} \text{Succ}_{\text{Supp}}(y_{i+1}, y_i) \wedge \bigwedge_{1 \leq i \leq N} P(y_i) = b_i^q \end{aligned}$$

where $P(y_i) = b_i^q$ is a shorthand for $P(y_i)$ if $b_i^q = \top$, and $\neg P(y_i)$ if $b_i^q = \perp$.

Encoding the Head Position

The position of the head is encoded in the second part of the grid, that is, in the interval $(3k+1, 3k+2)$ for the $(k+1)$ -th configuration (cf. Figure 3.1). The grid on this interval is order-isomorphic to $\mathbb{Z}$. Each element of this subordering will correspond to a position of the tape. When the predicate P is true at such a point, it means that the head points towards that cell. Since the Turing machines that we consider here have a single read/write head, it must point towards a unique cell for each configuration. Therefore P must be true only for a single element of that subordering.

Encoding the Tape Content

Similarly, the tape content is encoded in the third part of the grid, that is, in the interval $(3k + 2, 3k + 3)$ for the $(k + 1)$ -th configuration (cf. Figure 3.1). Again, the grid on this interval is order-isomorphic to $\mathbb{Z}$. And again, each element x of this subordering will correspond to a cell of the tape, matching the cell that corresponds to $x - 1$ in the head position interval. Figure 3.6 illustrates the connections between the suborderings, within a single configuration and with the next one. The idea of the encoding is to simply set the value of P to true on the elements of the subordering that correspond to cells containing a 1, and to false for cells containing a 0.

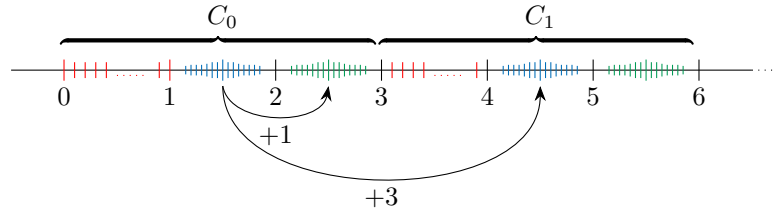


Figure 3.6: The first two consecutive configuration encodings.

3.2.5 Enforcing a Valid Run

Let us now define formally the formulas characterizing an accepting run of $\mathcal{M}$. We will decompose the global formula into three main parts: the initial conditions $START_{\mathcal{M}}$, the conditions on the transitions $STEP_{\mathcal{M}}$ and the halting condition $END_{\mathcal{M}}$. For the sake of clarity, we use capital letters for these higher-level formulas.

The initial conditions of $\mathcal{M}$ are that the state encoded on $\mathbf{0}$ and its $N - 1$ supporting successors is the initial state q_0 , that the head points towards a unique initial unspecified cell of the tape, and finally that the tape is initially filled with 0's. These conditions are expressed by the following formula:

$$\begin{aligned} START_{\mathcal{M}} = & State_{q_0}(\mathbf{0}) \wedge [\exists y. \mathbf{1} < y < \mathbf{2} \wedge Support(y) \wedge P(y) \\ & \wedge \forall x. (\mathbf{1} < x < \mathbf{2} \wedge Support(x) \wedge P(x)) \Rightarrow x = y] \\ & \wedge [\forall y. (\mathbf{2} < y < \mathbf{3} \wedge Support(y)) \Rightarrow \neg P(y)] \end{aligned}$$

The requirements on the transition are more complex. Intuitively, if before reaching the step $i \in \mathbb{N}$, we have not yet encountered the halting state q_F , then we must ensure that the configuration at Step i can be obtained from the configuration at the previous step $i - 1$ by following a transition $(q, \alpha, q', \alpha', \lambda) \in \Delta$. The overall formula for this condition is the following:

$$\begin{aligned} STEP_{\mathcal{M}} = & \forall y. (y > 0 \wedge EncodingBegins(y) \wedge NotEnded_{\mathcal{M}}(y)) \\ & \Rightarrow \exists x. y = x + 3 \wedge Transition_{\mathcal{M}}(x, y) \end{aligned}$$

The subformula $NotEnded_{\mathcal{M}}(y)$ expresses that no valid real value prior to y (i.e., a positive multiple of 3 strictly smaller than y) encodes the halting state. This formula is defined by:

$$NotEnded_{\mathcal{M}}(y) = \forall x. (x < y \wedge EncodingBegins(x)) \Rightarrow \neg(State_{q_F}(x))$$

The subformula $Transition_{\mathcal{M}}(x, y)$ expresses that there exists a transition $(q, \alpha, q', \alpha', \lambda) \in \Delta$ that allows to move in one step from the configuration encoded at x (i.e., that the encoding of the configuration starts exactly on x), to the configuration corresponding to y . To improve readability, we decompose the condition on the transition relation as follows:

$$Transition_{\mathcal{M}}(x, y) = \bigvee_{(q, \alpha, q', \alpha', \lambda) \in \Delta} \left[State_q(x) \wedge State_{q'}(y) \wedge Tape_{\alpha, \alpha'}(x, y) \wedge Head_{\lambda}(x, y) \right]$$

For a given transition $(q, \alpha, q', \alpha', \lambda) \in \Delta$, the conditions on the states, tape and head are expressed as follows:

- The state q must be encoded on the real value x , and the state q' on y : $State_q(x) \wedge State_{q'}(y)$
- The tape must contain $\alpha \in \{0, 1\}$ at the position of the head for the step corresponding to x . Additionally, for the step corresponding to y , the tape must contain α' at the previous position of the head, and remain unchanged at all other positions.

$$\begin{aligned} Tape_{\alpha, \alpha'}(x, y) &= [\forall z. (x + 1 < z < x + 2 \wedge Support(z) \wedge P(z)) \\ &\quad \Rightarrow P(z + 1) = \alpha \wedge P(z + 4) = \alpha'] \\ &\wedge [\forall z. (x + 1 < z < x + 2 \wedge Support(z) \wedge \neg P(z)) \Rightarrow (P(z + 1) \Leftrightarrow P(z + 4))] \end{aligned}$$

Where $P(z + k) = \alpha$ is a shorthand for $\exists u. u = z + k \wedge P(u)$ if $\alpha = 1$, and $\exists u. u = z + k \wedge \neg P(u)$ if $\alpha = 0$. The “+ 1” operator allows us to connect the encoding of the head position with the encoding of the tape content within the same configuration. The “+ 4” operator does the same while jumping to the next configuration (cf. Figure 4). Notice that this formula does not involve y ; it assumes (rightfully, given the formula $STEP_{\mathcal{M}}$) that the equality $y = x + 3$ holds.

- The head is moved in the direction specified by $\lambda \in \{L, R\}$, i.e., left for L and right for R . This can be expressed by exploiting the predecessor and successor relations defined for supporting real values.

$$\begin{aligned} Head_{\lambda}(x, y) &= \forall z. (x + 1 < z < x + 2 \wedge Support(z) \wedge P(z)) \\ &\quad \Rightarrow \exists v. f_{\lambda}(v, z + 3) \wedge P(v) \wedge \neg P(z) \end{aligned}$$

where $f_R = Succ_{Supp}$ and $f_L = Pred_{Supp}$. Since in the initial configuration of the Turing machine the head points towards a single cell, the formula $Head_\lambda$ ensures that this remains the case throughout every run of the Turing machine.

Finally, the existence of a halting run is expressed by the formula:

$$END_{\mathcal{M}} = \exists x. State_{q_F}(x)$$

The global formula that expresses that the Turing machine $\mathcal{M}$ halts on some run encoded by the value of the predicate P is the following:

$$HALT_{\mathcal{M}} = START_{\mathcal{M}} \wedge STEP_{\mathcal{M}} \wedge END_{\mathcal{M}} \wedge AXIOMS_{Supp}$$

where $AXIOMS_{Supp}$ is the axiomatization of the supporting points as described in Section 3.2.3.

By construction, satisfiability of the global formula $HALT_{\mathcal{M}}$ is equivalent to the existence of a halting run for the Turing machine $\mathcal{M}$. It follows that the satisfiability problem for $FO(\mathbb{R}, <, +1, P)$ is undecidable, which proves Theorem 3.2.1.

3.3 An Alternate Proof of the Decidability of $MFO(\mathbb{R}, <)$

In this section, we present a complete self-contained proof of decidability for the monadic first-order theory of order over real numbers $MFO(\mathbb{R}, <)$. The approach is similar to [73], but particularized to the special case of the real domain. Our construction also describes explicitly how to manage the uninterpreted monadic predicates involved in the logic, which is only hinted at in [73]. It relies on elements of the composition method, that are mainly borrowed from [114]. A corollary of the decidability result is that satisfiable formulas admit models that can be constructed in a simple way, by applying finitely many times composition operators to elementary sets. Additionally, we develop a symbolic notation system for such models, which can be seen as a first step towards an actual implementation of a decision procedure for the theory of order over the reals.

The remainder of this section reproduces the core — i.e., Sections 3 to 5 — of the following article [13]:

Bernard Boigelot, Pascal Fontaine, **Baptiste Vergain**. On the theory of order over real numbers. *Mathematics in Computer Science*, 20(1):5, 2026.

Our main interest is the domain $\mathbb{D} = \mathbb{R}$, but we will show in Section 3.3.3 that choosing $\mathbb{D}$ to be equal to any non-empty open interval (a, b) of $\mathbb{R}$ does not affect satisfiability, i.e., a formula is satisfiable in $MFO(\mathbb{R}, <)$ iff it is satisfiable in $MFO((a, b), <)$. We will exploit this property in order to simplify some developments.

3.3.1 Additional Definitions

For $n, m \geq 0$, a (n, m) -chain over the domain $\mathbb{D}$ is a tuple $(p_1, p_2, \dots, p_n, P_1, P_2, \dots, P_m)$, where $p_i \in \mathbb{D}$ for each $i \in [1, n]$, and $P_i \subseteq \mathbb{D}$ for each $i \in [1, m]$. The pair (n, m) is called the *dimension* of the chain. Intuitively, a (n, m) -chain is simply an interpretation that is not tied to a given formula, but that can be associated with any formula involving n free first-order variables, and m free second-order variables.

In the following, when we say that a formula $\varphi(x_1, x_2, \dots, x_n, X_1, X_2, \dots, X_m)$ with the free first-order variables $x_1, \dots, x_n$ and the free second-order variables $X_1, \dots, X_m$ is *satisfied* by the (n, m) -chain $\mathfrak{A} = (p_1, p_2, \dots, p_n, P_1, P_2, \dots, P_m)$, which is denoted by $\mathfrak{A} \models \varphi$, it should be understood as $\mathfrak{A}$ — viewed as an interpretation of φ over the structure $(\mathbb{D}, <)$ — being a model of φ .

3.3.2 Elements of Model Theory

The content of this section is mainly inspired by [114].

k -elementary equivalence

Two (n, m) -chains $\mathfrak{A}$ and $\mathfrak{B}$, with $n, m \geq 0$, are *k -elementarily equivalent* for some $k \geq 0$, which is denoted by $\mathfrak{A} \equiv_k \mathfrak{B}$, if for every formula $\varphi(x_1, \dots, x_n, X_1, \dots, X_m)$ of quantifier depth less than or equal to k , one has

$$\mathfrak{A} \models \varphi \Leftrightarrow \mathfrak{B} \models \varphi.$$

In other words, $\mathfrak{A}$ and $\mathfrak{B}$ cannot be distinguished by formulas with a quantifier depth at most equal to k .

Two (n, m) -chains $\mathfrak{A}$ and $\mathfrak{B}$ are *elementarily equivalent*, which is denoted by $\mathfrak{A} \equiv \mathfrak{B}$, if they are such that $\mathfrak{A} \equiv_k \mathfrak{B}$ for all $k \geq 0$.

k -types

A *k -type* for (n, m) -chains, with $k, n, m \geq 0$, is an equivalence class of the relation $\equiv_k$. In other words, two (n, m) -chains share the same k -type iff they cannot be distinguished by formulas with a quantifier depth that is at most equal to k .

A precise description of k -types can be constructed by induction on k . For the base case $k = 0$, we have to consider the equivalence classes of the relation $\equiv_0$. For given values of n and m , there are only finitely many distinct atoms over the first-order variables $x_1, x_2, \dots, x_n$ and the second-order variables $X_1, X_2, \dots, X_m$. Two (n, m) -chains $\mathfrak{A}$ and $\mathfrak{B}$ are such that $\mathfrak{A} \equiv_0 \mathfrak{B}$ iff the set of all atoms satisfied by $\mathfrak{A}$ is exactly equal to the set of all atoms satisfied by $\mathfrak{B}$. Indeed, in such a case, $\mathfrak{A}$ and $\mathfrak{B}$ cannot be distinguished by quantifier-free formulas, and otherwise one can easily build a quantifier-free formula satisfied by $\mathfrak{A}$ and not by $\mathfrak{B}$. More precisely, every 0-type τ for (n, m) -chains is described without ambiguity by its *characteristic formula* φ_τ , also known as *Hintikka*

formula [64, 43, 114], that is satisfied by a (n, m) -chain iff this chain belongs to τ . This formula takes the form

$$\varphi_\tau(x_1, \dots, x_n, X_1, \dots, X_m) = \bigwedge_{\psi \in \Lambda_\tau} \psi \wedge \bigwedge_{\psi \in \Lambda'_\tau} \neg\psi,$$

where Λ_τ and Λ'_τ are the sets of all atoms involving the first-order variables $x_1, x_2, \dots, x_n$ and the second-order variables $X_1, X_2, \dots, X_m$ that are respectively satisfied and not satisfied by the chains in τ . Note that the formula φ_τ is entirely characterized by the set of atoms Λ_τ ; this set thus provides a compact and unambiguous representation of the 0-type τ . Another important property is that, for given n and m , there are only finitely many 0-types for (n, m) -chains.

For the case $k > 0$, we have to characterize the equivalence classes of the relation $\equiv_k$. We show inductively that for every given n and m , each such equivalence class, i.e., each k -type τ for (n, m) -chains, is described without ambiguity by a Hintikka formula φ_τ , such that the (n, m) -chains that belong to τ are exactly those that satisfy φ_τ . Furthermore, for given k, n and m , there are finitely many such Hintikka formulas, thus there only exist finitely many k -types for (n, m) -chains.

Two (n, m) -chains $\mathfrak{A}$ and $\mathfrak{B}$ are such that $\mathfrak{A} \equiv_k \mathfrak{B}$ iff they cannot be distinguished by formulas of the form

$$Qx_{n+1} \varphi(x_1, x_2, \dots, x_{n+1}, X_1, X_2, \dots, X_m), \quad (3.5)$$

where $Q \in \{\exists, \forall\}$ and the quantifier depth of φ is $k-1$. By inductive hypothesis, all $(n+1, m)$ -chains $(p_1, p_2, \dots, p_{n+1}, P_1, P_2, \dots, P_m)$ are members of finitely many $(k-1)$ -types. Let $T = \{\tau_1, \tau_2, \dots, \tau_\ell\}$ be the set of these $(k-1)$ -types, and for every $i \in [1, \ell]$, let φ_i be the Hintikka formula describing τ_i .

For a (n, m) -chain $\mathfrak{A} = (p_1, p_2, \dots, p_n, P_1, P_2, \dots, P_m)$, each $(n+1, m)$ -chain $(p_1, p_2, \dots, p_{n+1}, P_1, P_2, \dots, P_m)$ that can be obtained by extending $\mathfrak{A}$ with an additional first-order value $p_{n+1} \in \mathbb{D}$ satisfies some Hintikka formula φ_i , with $i \in [1, \ell]$. (Note that we arbitrarily place the additional value p_{n+1} after the existing ones.) Let τ be the k -type of $\mathfrak{A}$, and I_τ be the set of all $i \in [1, \ell]$ such that $(p_1, p_2, \dots, p_{n+1}, P_1, P_2, \dots, P_m)$ satisfies φ_i for some $p_{n+1} \in \mathbb{D}$. The (n, m) -chains belonging to τ are then exactly those that satisfy the Hintikka formula

$$\begin{aligned} \varphi_\tau(x_1, \dots, x_n, X_1, \dots, X_m) = & \bigwedge_{i \in I_\tau} \exists x_{n+1} \varphi_i(x_1, \dots, x_{n+1}, X_1, \dots, X_m) \\ & \wedge \forall x_{n+1} \bigvee_{i \in I_\tau} \varphi_i(x_1, \dots, x_{n+1}, X_1, \dots, X_m), \end{aligned}$$

in which the first term expresses that each φ_i with $i \in I_\tau$ is satisfied for at least one value of p_{n+1} , and the second term imposes that for every value of p_{n+1} , there is some φ_i with $i \in I_\tau$ that is satisfied.

The Hintikka formula φ_τ provides a unambiguous representation of the k -type τ , since it admits the quantifier depth k and it is satisfied by a (n, m) -chain

iff this chain belongs to τ . Remark that the formula φ_τ is entirely characterized by the set I_τ , or equivalently by the set of $(k-1)$ -types T . This set T thus provides a compact and unambiguous representation of the k -type τ . In other words, a k -type, for $k > 0$, can uniquely be represented as a set of $(k-1)$ -types, of which there exist only finitely many instances. We have just established the following theorem.

Theorem 3.3.1. *For given values of $k, n, m \geq 0$, there exist finitely many k -types for (n, m) -chains.*

From now on, we will assimilate k -types with their descriptions, i.e., we will consider that a k -type corresponds indifferently to a set of chains belonging to some equivalence class of $\equiv_k$, to a Hintikka formula representing this class, or to a set of either atoms (if $k = 0$), or $(k-1)$ -types (if $k > 0$). In most technical developments, we will rely on the representation of types as sets.

Note that a k -type is also a $(k-1)$ -type, and the description of this $(k-1)$ -type as a set can easily be computed recursively. Indeed, for $k = 1$, the 0-type corresponding to a given 1-type τ is simply the same 0-type obtained by suppressing in any 0-type belonging to τ all atoms referring to the last first-order variable. For $k > 1$, the $(k-1)$ -type for a given k -type τ is the set of $(k-2)$ -types corresponding to the $(k-1)$ -types that belong to τ .

Deciding Formulas with Types

Let $\mathfrak{A}$ be a (n, m) -chain, with $n, m \geq 0$, and φ be a formula of $\text{MFO}(\mathbb{D}, <)$ with n first-order and m second-order free variables. Let k be the quantifier depth of φ . From the k -type of $\mathfrak{A}$, one can decide whether $\mathfrak{A} \models \varphi$ holds or not.

Indeed, for the case $k = 0$, the 0-type of $\mathfrak{A}$ corresponds to the set of all the atoms involving n first-order and m second-order variables that are satisfied by $\mathfrak{A}$. Deciding the quantifier-free formula φ then amounts to evaluating the truth value of this formula when each such atom is replaced by $\top$, and all the others by $\perp$.

If $k > 0$, then the k -type τ of $\mathfrak{A}$ corresponds to the set of $(k-1)$ -types of the $(n+1, m)$ -chains obtained by extending $\mathfrak{A}$ with one additional first-order value $p_{n+1} \in \mathbb{D}$, for all possible choices of p_{n+1} . If φ is of the form

$$\exists x_{n+1} \psi(x_1, x_2, \dots, x_{n+1}, X_1, X_2, \dots, X_m)$$

where the quantifier depth of ψ is $k-1$, then deciding whether $\mathfrak{A} \models \varphi$ holds reduces to checking whether ψ is satisfied by the chains that belong to at least one of the $(k-1)$ -types that belong to τ . (Recall that all the chains in such a $(k-1)$ -type satisfy the same formulas of quantifier depth $k-1$.) There are finitely many such types, hence this decision procedure can be carried out by induction on k . Similarly, if φ is of the form

$$\forall x_{n+1} \psi(x_1, x_2, \dots, x_{n+1}, X_1, X_2, \dots, X_m)$$

then deciding $\mathfrak{A} \models \varphi$ amounts to checking that ψ is satisfied by the chains that belong to every $(k-1)$ -type that belongs to τ .

3.3.3 Constructible Chains

We now discuss how to construct a particular class of chains by starting from basic building blocks and combining them with the help of composition operators. Before we introduce those elements and operators, we first show that the domain over which a chain is considered does not impact its k -type.

Domain Equivalence

Let $\mathbb{D}_1 = (a_1, b_1)$ and $\mathbb{D}_2 = (a_2, b_2)$ be two domains, with $a_1, a_2 \in \mathbb{R} \cup \{-\infty\}$, $b_1, b_2 \in \mathbb{R} \cup \{+\infty\}$, $a_1 < b_1$ and $a_2 < b_2$. We denote by $\mathfrak{A}_i \models_i \varphi$ the fact that a chain $\mathcal{A}_i$ over $\mathbb{D}_i$ satisfies a formula φ in $\text{MFO}(\mathbb{D}_i, <)$.

Let f be a strictly increasing continuous total function from $\mathbb{D}_1$ to $\mathbb{D}_2$, i.e., such that $f(\mathbb{D}_1) = \mathbb{D}_2$, and $f(x) < f(y)$ for every $x, y \in \mathbb{D}_1$ such that $x < y$. This function transforms any (n, m) -chain $\mathfrak{A}_1 = (p_{1,1}, p_{1,2}, \dots, p_{1,n}, P_{1,1}, P_{1,2}, \dots, P_{1,m})$ over $\mathbb{D}_1$ into a (n, m) -chain $\mathfrak{A}_2 = (p_{2,1}, p_{2,2}, \dots, p_{2,n}, P_{2,1}, P_{2,2}, \dots, P_{2,m})$ over $\mathbb{D}_2$, where $p_{2,i} = f(p_{1,i})$ for every $i \in [1, n]$, and $P_{2,i} = \{f(p) \mid p \in P_{1,i}\}$ for every $i \in [1, m]$.

Theorem 3.3.2. *For any such f , $\mathfrak{A}_1$, $\mathfrak{A}_2$ and formula φ , we have*

$$\mathfrak{A}_1 \models_1 \varphi \Leftrightarrow \mathfrak{A}_2 \models_2 \varphi.$$

In other words, $\mathfrak{A}_1$ and $\mathfrak{A}_2$ share the same k -type for every $k \geq 0$.

Proof. Let k be the quantifier depth of φ . If $k = 0$, then $\mathfrak{A}_1$ and $\mathfrak{A}_2$ satisfy exactly the same subset of atoms, hence they share the same 0-type, and $\mathfrak{A}_1 \models_1 \varphi \Leftrightarrow \mathfrak{A}_2 \models_2 \varphi$. If $k > 0$, then one establishes by induction on k that the k -types of $\mathfrak{A}_1$ and $\mathfrak{A}_2$ are identical by construction, hence $\mathfrak{A}_1 \models_1 \varphi \Leftrightarrow \mathfrak{A}_2 \models_2 \varphi$ as well. $\square$

The following corollary shows that the choice of domain does not influence the satisfiability of formulas in $\text{MFO}(\mathbb{D}, <)$, where $\mathbb{D}$ is any non-empty open interval of $\mathbb{R}$.

Corollary 3.3.1. *A formula φ is satisfiable in $\text{MFO}(\mathbb{D}_1, <)$ iff it is satisfiable in $\text{MFO}(\mathbb{D}_2, <)$.*

Proof. For every pair of domains $\mathbb{D}_1$ and $\mathbb{D}_2$, there exists a strictly increasing continuous total function f from $\mathbb{D}_1$ to $\mathbb{D}_2$. The result then follows from Theorem 3.3.2. $\square$

Elementary Chains

We now describe the basic building blocks that will be used for constructing chains, and introduce a symbolic notation for them. The idea is that each instance of this notation describes a set of chains that share the same k -type (in other words, that are k -elementarily equivalent) for every $k \geq 0$. As explained previously, the domain $\mathbb{D}$ over which chains are defined can be chosen equal to any non-empty open interval of $\mathbb{R}$.

Constant Chains Let $m \geq 0$. The notation $\llbracket U \rrbracket$, with $U \subseteq \{1, 2, \dots, m\}$, describes a set containing a single $(0, m)$ -chain $(P_1, P_2, \dots, P_m)$ such that for every $i \in [1, m]$, $P_i = \mathbb{D}$ if $i \in U$ and $P_i = \emptyset$ if $i \notin U$. In other words, the values i in U characterize the predicates P_i that are true over the whole domain $\mathbb{D}$, while the other predicates are false everywhere.

Shuffle Chains Let $m \geq 0$ and $\ell > 0$. The notation $\llbracket U_1, U_2, \dots, U_\ell \rrbracket$, with $U_i \subseteq \{1, 2, \dots, m\}$ for every $i \in [1, \ell]$, describes the set of all the $(0, m)$ -chains $(P_1, P_2, \dots, P_m)$ that satisfy the two following properties:

- For each $p \in \mathbb{D}$, there exists $i \in [1, \ell]$ such that $U_i = \{j \in [1, m] \mid p \in P_j\}$. This means that the components U_i represent all the combinations of predicate values that occur over the elements of $\mathbb{D}$.
- For all $i \in [1, \ell]$ and $p_1, p_2 \in \mathbb{D}$ such that $p_1 < p_2$, there exists p such that $p_1 < p < p_2$ and $U_i = \{j \in [1, m] \mid p \in P_j\}$. In other words, for every bounded open interval in $\mathbb{D}$ and component U_i of the expression, there must exist a point p in this interval at which the combination of predicate values is represented by U_i .

Note that in the particular case where $\ell = 1$, this definition corresponds to the constant chains introduced in Section 3.3.3. This is the reason why the two notations coincide in this case.

Theorem 3.3.3. *Let $k \geq 0$. All the chains contained in the set $\llbracket U_1, U_2, \dots, U_\ell \rrbracket$ share the same k -type.*

Proof. We actually prove a stronger property: Consider two (n, m) -chains $\mathfrak{A}_1 = (p_{1,1}, p_{1,2}, \dots, p_{1,n}, P_{1,1}, P_{1,2}, \dots, P_{1,m})$ and $\mathfrak{A}_2 = (p_{2,1}, p_{2,2}, \dots, p_{2,n}, P_{2,1}, P_{2,2}, \dots, P_{2,m})$, with $n \geq 0$, such that

- $(P_{1,1}, P_{1,2}, \dots, P_{1,m}) \in \llbracket U_1, U_2, \dots, U_\ell \rrbracket$,
- $(P_{2,1}, P_{2,2}, \dots, P_{2,m}) \in \llbracket U_1, U_2, \dots, U_\ell \rrbracket$,
- $p_{1,i} \in P_{1,j} \Leftrightarrow p_{2,i} \in P_{2,j}$ for all $i \in [1, n]$ and $j \in [1, m]$, and
- $p_{1,i} \leq p_{1,j} \Leftrightarrow p_{2,i} \leq p_{2,j}$ for all $i, j \in [1, n]$.

We will establish that, under these hypotheses, $\mathfrak{A}_1$ and $\mathfrak{A}_2$ share the same k -type for every $k \geq 0$. The theorem then corresponds to the special case $n = 0$ of this property.

The proof is by induction on k . For the base case $k = 0$, the (n, m) -chains $\mathfrak{A}_1$ and $\mathfrak{A}_2$ satisfy exactly the same atoms, thus they have the same 0-type. For the inductive case $k > 0$, we know that the k -type of $\mathfrak{A}_1$ corresponds to the set of $(k-1)$ -types of all the $(n+1, m)$ -chains $(p_{1,1}, p_{1,2}, \dots, p_{1,n+1}, P_{1,1}, P_{1,2}, \dots, P_{1,m})$ that can be obtained by extending $\mathfrak{A}_1$ with one additional first-order value $p_{1,n+1} \in \mathbb{D}$. Likewise, the k -type of $\mathfrak{A}_2$ corresponds to the set of $(k-1)$ -types of all the $(n+1, m)$ -chains $(p_{2,1}, p_{2,2}, \dots, p_{2,n+1}, P_{2,1}, P_{2,2}, \dots, P_{2,m})$ obtained by extending $\mathfrak{A}_2$ with $p_{2,n+1} \in \mathbb{D}$.

It is sufficient to show that for each choice of $p_{1,n+1} \in \mathbb{D}$, there exists $p_{2,n+1} \in \mathbb{D}$ such that the $(k-1)$ -types of $(p_{1,1}, p_{1,2}, \dots, p_{1,n+1}, P_{1,1}, P_{1,2}, \dots, P_{1,m})$ and $(p_{2,1}, p_{2,2}, \dots, p_{2,n+1}, P_{2,1}, P_{2,2}, \dots, P_{2,m})$ are identical. (By symmetry of the problem, this implies that for each choice of $p_{2,n+1} \in \mathbb{D}$, there exists a matching $p_{1,n+1} \in \mathbb{D}$ as well.)

Consider a particular value of $p_{1,n+1} \in \mathbb{D}$. Either this value is equal to some other $p_{1,i}$, with $i \in [1, n]$, or it is not. In the former case, picking $p_{2,n+1} = p_{2,i}$ entails that the chains $(p_{1,1}, p_{1,2}, \dots, p_{1,n+1}, P_{1,1}, P_{1,2}, \dots, P_{1,m})$ and $(p_{2,1}, p_{2,2}, \dots, p_{2,n+1}, P_{2,1}, P_{2,2}, \dots, P_{2,m})$ have the same $(k-1)$ -type by inductive hypothesis.

If, on the other hand, $p_{1,n+1} \neq p_{1,i}$ for every $i \in [1, n]$, then there exists an open interval (α_1, β_1) of $\mathbb{D}$, with $\alpha_1 \in \{-\infty, p_{1,1}, p_{1,2}, \dots, p_{1,n}\}$, $\beta_1 \in \{p_{1,1}, p_{1,2}, \dots, p_{1,n}, +\infty\}$ and $\alpha_1 < \beta_1$, such that this interval contains $p_{1,n+1}$ and does not contain any $p_{1,i}$ with $i \in [1, n]$. We then construct the matching interval (α_2, β_2) by setting $\alpha_2 \in \{-\infty, p_{2,1}, p_{2,2}, \dots, p_{2,n}\}$ and $\beta_2 \in \{p_{2,1}, p_{2,2}, \dots, p_{2,n}, +\infty\}$ so that $\alpha_2 = p_{2,i}$ iff $\alpha_1 = p_{1,i}$, and $\beta_2 = p_{2,i}$ iff $\beta_1 = p_{1,i}$, for every $i \in [1, n]$. We then pick $p_{2,n+1}$ in the interval (α_2, β_2) , such that the value of $(P_{2,1}, P_{2,2}, \dots, P_{2,m})$ at $p_{2,n+1}$ is identical to the value of $(P_{1,1}, P_{1,2}, \dots, P_{1,m})$ at $p_{1,n+1}$. This is always possible by definition of shuffle chains, since both $(P_{1,1}, P_{1,2}, \dots, P_{1,m})$ and $(P_{2,1}, P_{2,2}, \dots, P_{2,m})$ belong to $\llbracket U_1, U_2, \dots, U_\ell \rrbracket$; this implies that the combination of predicate values at each point corresponds to some set U_i , and that each U_i occurs at least once in every bounded open interval of $\mathbb{D}$.

With such a choice of value for $p_{2,n+1}$, the chains $(p_{1,1}, p_{1,2}, \dots, p_{1,n+1}, P_{1,1}, P_{1,2}, \dots, P_{1,m})$ and $(p_{2,1}, p_{2,2}, \dots, p_{2,n+1}, P_{2,1}, P_{2,2}, \dots, P_{2,m})$ have the same $(k-1)$ -type by inductive hypothesis, which concludes the proof. $\square$

It may not be obvious that the set of $(0, m)$ -chains described by $\llbracket U_1, \dots, U_\ell \rrbracket$ is not empty. We now show how to explicitly build such a chain $(P_1, P_2, \dots, P_m)$, over the domain $\mathbb{D} = (0, 1)$, by a Cantor-set-like construction. First, we split evenly the interval $(0, 1)$ into ℓ parts by setting $p_i = i/\ell$ for each $i \in [1, \ell-1]$. We then fix, for each $i \in [1, \ell-1]$ and $j \in [1, m]$, $P_j(p_i) = \top$, i.e., $p_i \in P_j$, if $j \in U_i$, and $P_j(p_i) = \perp$ if $j \notin U_i$. Then, in every real interval $(0, p_1), (p_1, p_2), \dots, (p_{\ell-2}, p_{\ell-1}), (p_{\ell-1}, 1)$ delimited by the points p_i , we carry out recursively the same operation. Finally, for each $p \in (0, 1)$ for which the value of the predicates P_j is still undefined, we set for each $j \in [1, m]$, $P_j(p) = \top$ if $j \in U_\ell$ and $P_j(p) = \perp$ if $j \notin U_\ell$. As a side remark, notice that the particular chain built by this process has countably many elements corresponding to each U_i for each $i \in [1, \ell-1]$ and an uncountable number of elements corresponding to U_ℓ . This is of no consequence in our context.

Type Computation Let $m \geq 0$, $\ell > 0$ and $U_i \subseteq \{1, 2, \dots, m\}$ for every $i \in [1, \ell]$. We now show that for every $k \geq 0$, the k -type of the $(0, m)$ -chains in $\llbracket U_1, U_2, \dots, U_\ell \rrbracket$ can be effectively computed.

Like in the proof of Theorem 3.3.3, we consider the more general case of (n, m) -chains $(p_1, p_2, \dots, p_n, P_1, P_2, \dots, P_m)$, with $n \geq 0$, such that

- $(P_1, P_2, \dots, P_m) \in \llbracket U_1, U_2, \dots, U_\ell \rrbracket$, and
- for each $i \in [1, n]$ and $j \in [1, m]$, the Boolean value $t_{i,j}$ such that $t_{i,j} = \top$ iff $p_i \in P_j$ is fixed and known.

We show how to compute the k -type of such chains, given $U_1, U_2, \dots, U_\ell, p_1, p_2, \dots, p_n$, and $t_{i,j}$ for all $i \in [1, n]$ and $j \in [1, m]$. The original problem then corresponds to the particular case where $n = 0$ of this more general setting.

For the case $k = 0$, we have to compute the set of all the atoms involving n first-order and m second-order variables that are satisfied by the considered chains. This is immediate: the truth values of atoms of the form $x_i = x_j$ and $x_i < x_j$ are obtained from the values of $p_1, p_2, \dots, p_n$, and those of the atoms $x_i \in X_j$ are provided by the truth values $t_{i,j}$.

For the case $k > 0$, we have to compute the set of $(k - 1)$ -types of all the $(n + 1, m)$ -chains that can be obtained by adding one new first-order value $p_{n+1} \in \mathbb{D}$, for all possible choices of this value. The possible choices of p_{n+1} are

- to make p_{n+1} equal to some p_i with $i \in [1, n]$, if $n > 0$. In this case, we have $t_{n+1,j} = t_{i,j}$ for all $j \in [1, m]$.
- to make p_{n+1} different from all the p_i with $i \in [1, n]$ (this case allows $n = 0$). In this case,
 - there are only finitely many classes of values for p_{n+1} that can be distinguished by atoms of the form $x_i = x_j$ and $x_i \leq x_j$ (evaluated over all the p_i for $i \in [1, n + 1]$).
 - the values of $t_{n+1,j}$, for $j \in [1, m]$ can correspond to any of the U_i with $i \in [1, \ell]$. More precisely, for every choice of $i \in [1, \ell]$, one can set the $t_{n+1,j}$, for each $j \in [1, m]$, so that $t_{n+1,j} = \top$ iff $j \in U_i$.

We have established that there are only finitely many choices of p_{n+1} and $t_{n+1,j}$ for every $j \in [1, m]$ that can lead to different $(k - 1)$ -types for the $(n + 1, m)$ -chains obtained by adding the new first-order value p_{n+1} . One can then explicitly enumerate those choices, and compute the corresponding $(k - 1)$ -types by induction on k .

Composition Operators

We now introduce operators that allow to transform and combine the sets of chains that can be described by our notation. For each of them, we define how the composite chains are constructed from their underlying components, and show that chains sharing identical notations always have the same k -type for all $k \geq 0$. Most proofs in this section are concrete applications of the composition method mentioned in Chapter 2.

First-Order Variables Reordering Let S be a set of (n, m) -chains, with $n, m \geq 0$, and $V = [i_1; i_2; \dots; i_n]$ be a permutation of $[1; 2; \dots; n]$. The notation

$$S/V$$

describes the set of all the chains contained in S in which the first-order values are reordered according to V . Formally, S/V is the set of all the chains $(p_{i_1}, p_{i_2}, \dots, p_{i_n}, P_1, P_2, \dots, P_m)$ such that $(p_1, p_2, \dots, p_n, P_1, P_2, \dots, P_m) \in S$. In other words, each first-order variable x_j involved in the expression of S , with $j \in [1, n]$, is renamed x_{i_j} .

Theorem 3.3.4. *Let $k \geq 0$. All the chains contained in S/V share the same k -type, assuming that the same property holds for S .*

Proof. Consider two (n, m) -chains $\mathfrak{A}$ and $\mathfrak{B}$ belonging to the set described by S/V . Let us show that these two chains share the same k -type for any $k \geq 0$. By definition of the reordering operator, the chains $\mathfrak{A}$ and $\mathfrak{B}$ correspond respectively to (n, m) -chains $\mathfrak{A}_0$ and $\mathfrak{B}_0$ of S , in which the first-order values have been reordered according to V .

The proof is by induction on k . For the base case $k = 0$, the sets of atoms satisfied by $\mathfrak{A}$ and $\mathfrak{B}$ correspond respectively to the atoms satisfied by the (n, m) -chains $\mathfrak{A}_0$ and $\mathfrak{B}_0$ in which the first-order variables are renamed according to V . Since those sets of atoms are identical for $\mathfrak{A}_0$ and $\mathfrak{B}_0$ by hypothesis, they are identical for $\mathfrak{A}$ and $\mathfrak{B}$ as well after the renaming operation.

For the inductive case $k > 0$, the k -type τ of $\mathfrak{A}$ corresponds to the set of $(k-1)$ -types of the (n, m) -chains obtained by extending $\mathfrak{A}$ with one additional first-order value p_{n+1} , for all possible choices of this value. Such chains take the form $(p_{i_1}, p_{i_2}, \dots, p_{i_n}, p_{n+1}, P_1, P_2, \dots, P_m)$, where $(p_1, p_2, \dots, p_n, p_{n+1}, P_1, P_2, \dots, P_m)$ is an element of the $(k-1)$ -type of $\mathfrak{A}_0$. In other words, the elements of the k -type of the elements of S/V can be obtained by applying the variable reordering induced by the permutation $[i_1; i_2; \dots; i_n; n+1]$ to the elements of the (supposed unique) k -type of the elements of S . By inductive hypothesis, the result of this operation is identical for all elements of S , in particular for $\mathfrak{A}$ and $\mathfrak{B}$, which therefore admit the same k -type. $\square$

Concatenation Let S_1 and S_2 be sets of (respectively) (n_1, m) and (n_2, m) -chains, with $n_1, n_2, m \geq 0$. Thanks to Theorem 3.3.2, we can freely assign the domains of the chains in S_1 and S_2 to be any non-empty open interval of $\mathbb{R}$ without affecting their type. We fix the domain $(-\infty, 0)$ for S_1 and the domain $(0, +\infty)$ for S_2 . The notation

$$S_1 +_U^\ell S_2,$$

with $\ell \geq 0$ and $U \subseteq \{1, 2, \dots, m\}$, then describes the set of all the $(n_1 + \ell + n_2, m)$ -chains $(p_1, p_2, \dots, p_{n_1 + \ell + n_2}, P_1, P_2, \dots, P_m)$ over the domain $\mathbb{R}$ such that

- the chain $(p_1, p_2, \dots, p_{n_1}, P'_1, P'_2, \dots, P'_m)$ belongs to S_1 , where for each $i \in [1, m]$, P'_i is the restriction of P_i to the domain $(-\infty, 0)$, i.e., $P'_i = P_i \cap (-\infty, 0)$.

- $p_{n_1+i} = 0$ for each $i \in [1, \ell]$.
- the chain $(p_{n_1+\ell+1}, \dots, p_{n_1+\ell+n_2}, P_1'', P_2'', \dots, P_m'')$ belongs to S_2 , where for each $i \in [1, m]$, P_i'' is the restriction of P_i to the domain $(0, +\infty)$, i.e., $P_i'' = P_i \cap (0, +\infty)$.
- for each $i \in [1, m]$, $0 \in P_i$ iff $i \in U$.

In other words, the domain is split into two parts. The values of the predicates in the first part are described by S_1 together with those of the first n_1 first-order variables, and in the second part by S_2 with the n_2 last first-order variables. At the junction point between the two parts, there are ℓ additional first-order variables, and the values of the predicates at this point are specified by U . In this definition, the junction point is fixed at 0 for the sake of simplicity, but any other real value could indifferently be used, since by Theorem 3.3.2 the domains can be transformed at will. Note that the additional first-order variables are inserted after those from S_1 and before those from S_2 , in order to make the notation associative: the chains described by $(S_1 +_{U_1}^{\ell_1} S_2) +_{U_2}^{\ell_2} S_3$ are identical to those described by $S_1 +_{U_1}^{\ell_1} (S_2 +_{U_2}^{\ell_2} S_3)$, and one can simply write $S_1 +_{U_1}^{\ell_1} S_2 +_{U_2}^{\ell_2} S_3$.

In the particular case where no additional first-order variable is inserted, we write $S_1 +_U S_2$ as shorthand for $S_1 +_U^0 S_2$.

Theorem 3.3.5. *Let $k \geq 0$. All the chains contained in $S_1 +_U^\ell S_2$ share the same k -type, assuming that the same property holds for S_1 and S_2 .*

Proof. W.l.o.g., we assume that the domains assigned to the chains belonging to S_1 , S_2 and $S_1 +_U^\ell S_2$ are respectively $(-\infty, 0)$, $(0, +\infty)$ and $\mathbb{R}$.

Consider two $(n_1 + \ell + n_2, m)$ -chains $\mathfrak{A}$ and $\mathfrak{B}$ belonging to the set described by $S_1 +_U^\ell S_2$. Let us show that these two chains share the same k -type for any $k \geq 0$. The proof is by induction on k . For the base case $k = 0$, we inspect the atoms satisfied by $\mathfrak{A}$ or $\mathfrak{B}$. The set of these atoms is composed of

- the atom $\top$.
- the atoms satisfied by any (n_1, m) -chain $(p_1, p_2, \dots, p_{n_1}, P_1', P_2', \dots, P_m')$ belonging to S_1 . Recall that by hypothesis, all such chains share the same 0-type, thus the set of such atoms does not depend on the particular chain being chosen.
- the atoms satisfied by any (n_2, m) -chain $(p_1, p_2, \dots, p_{n_2}, P_1'', P_2'', \dots, P_m'')$ belonging to S_2 , in which the first-order variables $x_1, x_2, \dots, x_{n_2}$ are (respectively) renamed $x_{n_1+\ell+1}, x_{n_1+\ell+2}, \dots, x_{n_1+\ell+n_2}$. (The same remark applies.)
- the atoms $x_{i_1} < x_{i_2}$, $x_{i_1} < x_j$ and $x_j < x_{i_2}$ for every $i_1 \in [1, n_1]$, $i_2 \in [n_1 + \ell + 1, n_1 + \ell + n_2]$ and $j \in [n_1 + 1, n_1 + \ell]$.
- the atoms $x_i \in X_j$ for every $i \in [n_1 + 1, n_1 + \ell]$ and $j \in U$.

This set of atoms is thus identical for $\mathfrak{A}$ and $\mathfrak{B}$, hence these two chains share the same 0-type.

For the inductive case $k > 0$, we have to characterize the set T of $(k-1)$ -types of the $(n_1 + \ell + n_2 + 1, m)$ -chains obtained by extending $\mathfrak{A}$ and $\mathfrak{B}$ with an additional first-order value $p_{n_1+\ell+n_2+1} \in \mathbb{R}$, for all possible choices of this value. There are several possibilities:

- $p_{n_1+\ell+n_2+1} < 0$. Then, each element τ of the k -type of the chains in S_1 (supposed to be unique by hypothesis) provides the $(k-1)$ -type of some set S_1^τ of $(n_1 + 1, m)$ -chains obtained by adding such a first-order value $p_{n_1+\ell+n_2+1}$ to a (n_1, m) -chain in S_1 . For each S_1^τ obtained in this way, the $(n_1 + \ell + n_2 + 1)$ -chains described by $S_1^\tau +_U^\ell S_2$ share the same $(k-1)$ -type by inductive hypothesis. By Theorem 3.3.4, the $(n_1 + \ell + n_2 + 1)$ -chains described by $(S_1^\tau +_U^\ell S_2)/V$, with $V = [1; 2; \dots; n_1; n_1 + 2; n_1 + 3; \dots; n_1 + \ell + n_2 + 1; n_1 + 1]$ share the same $(k-1)$ -type as well. This $(k-1)$ -type is an element of T .
- $p_{n_1+\ell+n_2+1} > 0$. Similarly, each element τ of the k -type of the chains in S_2 provides the $(k-1)$ -type of some set S_2^τ of $(n_2 + 1, m)$ -chains obtained by adding a first-order value to a (n_2, m) -chain in S_2 . For each S_2^τ obtained in this way, the $(n_1 + \ell + n_2 + 1)$ -chains described by $S_1 +_U^\ell S_2^\tau$ share the same $(k-1)$ -type by inductive hypothesis. This $(k-1)$ -type is an element of T .
- $p_{n_1+\ell+n_2+1} = 0$. In this case, the $(n_1 + \ell + n_2 + 1, m)$ -chains obtained by extending $\mathfrak{A}$ and $\mathfrak{B}$ with $p_{n_1+\ell+n_2+1}$ belong to the set described by $(S_1 +_U^{(\ell+1)} S_2)/V$, where $V = [1; 2; \dots; n_1 + \ell; n_1 + \ell + 2; n_1 + \ell + 3; \dots; n_1 + \ell + n_2 + 1; n_1 + \ell + 1]$. By inductive hypothesis, the $(n_1 + \ell + n_2 + 1, m)$ -chains described by $S_1 +_U^{(\ell+1)} S_2$ share the same $(k-1)$ -type. By Theorem 3.3.4, the $(n_1 + \ell + n_2 + 1, m)$ -chains described by $(S_1 +_U^{(\ell+1)} S_2)/V$ share the same $(k-1)$ -type as well. This $(k-1)$ -type is an element of T .

The set T of all the $(k-1)$ -types obtained in this way is finite, and corresponds to the k -types of $\mathfrak{A}$ and $\mathfrak{B}$. These k -types are therefore identical. $\square$

Note that the proof of Theorem 3.3.5 provides an algorithm for computing the k -type of the chains described by $S_1 +_U^\ell S_2$ from the k -types of those described by S_1 and S_2 , for any $k \geq 0$.

Omega Repetition Let S be a set of $(0, m)$ -chains, with $m \geq 0$. The notation

$$S_U^{+\omega},$$

with $U \subseteq \{1, 2, \dots, m\}$ describes the set of all the $(0, m)$ -chains that can be transformed by Theorem 3.3.2 into chains $(P_1, P_2, \dots, P_m)$ over the domain $(0, +\infty)$ such that

- for each $j \in \mathbb{N}$, the chain $(P'_1, P'_2, \dots, P'_m)$ belongs to S , where for each $i \in [1, m]$, P'_i is the restriction of P_i to the open interval $(j, j+1)$, i.e., $P'_i = P_i \cap (j, j+1)$. Note that this definition assigns the domain $(j, j+1)$ to the chain $(P'_1, P'_2, \dots, P'_m)$, but thanks to Theorem 3.3.2, we can w.l.o.g. transform this domain so that it matches the domain of the chains in S .
- for each $j \in \mathbb{N}_{>0}$ and $i \in \{1, 2, \dots, m\}$, we have $j \in P_i$ iff $i \in U$.

In other words, the domain is split into countably many successive open intervals, with an initial one. Inside every such interval, the values of the predicates are described by S . At the junction points between the intervals, the values of the predicates are specified by U .

Theorem 3.3.6. *Let $k \geq 0$. All the chains contained in $S_U^{+\omega}$ share the same k -type, assuming that the same property holds for S .*

Proof. We will prove a stronger property. Let S_1 be a set of (n, m) -chains that all share the same k -type for every $k \geq 0$, with $n, m \geq 0$, and let S_2 be a set of $(0, m)$ -chains that share the same k -type for every $k \geq 0$. (The types of the chains in S_1 and S_2 can be different.) Let $U \subseteq \{1, 2, \dots, m\}$. We show that all the (n, m) -chains belonging to the set described by

$$S_1 +_U (S_2)_U^{+\omega}$$

share the same k -type for every $k \geq 0$. The statement of the theorem then corresponds to the particular case where $S_1 = S_2 = S$.

We assume w.l.o.g. that the domains of S_1 and $(S_2)_U^{+\omega}$ are respectively $(-\infty, 0)$ and $(0, +\infty)$. Furthermore, we suppose that the $(0, m)$ -chains that belong to $(S_2)_U^{+\omega}$ match chains in S_2 over each open interval $(j, j+1)$ with $j \in \mathbb{N}$, i.e., that the junction points for the $\cdot_U^{+\omega}$ operator are fixed at the strictly positive integers.

Let $\mathfrak{A}$ and $\mathfrak{B}$ be two (n, m) -chains belonging to the set described by $S_1 +_U (S_2)_U^{+\omega}$. We show by induction on k that $\mathfrak{A}$ and $\mathfrak{B}$ have the same k -type.

For the base case $k = 0$, we need to characterize the set of atoms satisfied by $\mathfrak{A}$ and $\mathfrak{B}$. Since these atoms either correspond to $\top$ or involve a first-order variable, they necessarily belong to the 0-type of S_1 . They are thus identical for $\mathfrak{A}$ and $\mathfrak{B}$, hence these two (n, m) -chains have the same 0-type.

For the inductive case $k > 0$, we construct the set T of $(k-1)$ -types of the $(n+1, m)$ -chains obtained by extending $\mathfrak{A}$ and $\mathfrak{B}$ with an additional first-order value $p_{n+1} \in \mathbb{R}$, for all possible choices of this value. There are several possibilities:

- $p_{n+1} < 0$. Then, each element τ of the k -type of the chains in S_1 (supposed to be unique by hypothesis) provides the $(k-1)$ -type of some set S_1^τ of $(n+1, m)$ -chains obtained by adding such a first-order value p_{n+1} to a (n, m) -chain in S_1 . For each S_1^τ obtained in this way, the $(n+1, m)$ -chains described by $S_1^\tau +_U (S_2)_U^{+\omega}$ share the same $(k-1)$ -type by inductive hypothesis. This $(k-1)$ -type is an element of T .

- $j < p_{n+1} < j + 1$ for some $j \in \mathbb{N}$. Let us assume (w.l.o.g. thanks to Theorem 3.3.2) that the $(0, m)$ -chains in S_2 are defined over the domain $(j, j+1)$. The k -type of these chains is unique by hypothesis. Each element τ of this k -type provides the $(k-1)$ -type of some set S_2^τ of $(1, m)$ -chains obtained by adding a first-order value p_{n+1} to a $(0, m)$ -chain in S_2 . For each S_2^τ obtained in this way, the $(n+1, m)$ -chains described by

$$S_3^{\tau, j} = S_1 +_U S_2 +_U S_2 +_U \cdots +_U S_2 +_U S_2^\tau,$$

where there are $j+1$ occurrences of the operator $+_U$, share the same $(k-1)$ -type by Theorem 3.3.5. Therefore, by inductive hypothesis, the $(n+1, m)$ -chains described by

$$S_3^{\tau, j} +_U (S_2)_U^{+\omega}$$

share the same $(k-1)$ -type. This $(k-1)$ -type is an element of T .

- $p_{n+1} = j$ for some $j \in \mathbb{N}$. In this case, the $(n+1, m)$ -chains obtained by extending $\mathfrak{A}$ and $\mathfrak{B}$ with p_{n+1} belong to the set described by

$$S_4^j +_U (S_2)_U^{+\omega}, \quad (3.6)$$

with

$$S_4^j = S_1 +_U S_2 +_U S_2 +_U \cdots +_U S_2 +_{\frac{1}{U}} S_2,$$

where there are $j+1$ occurrences of $+_U$ (counting $+_{\frac{1}{U}}$ as well). By Theorem 3.3.5, all the $(n+1, m)$ -chains described by S_4^j share the same $(k-1)$ -type, therefore by inductive hypothesis, the $(n+1, m)$ -chains described by (3.6) share the same $k-1$ type as well. This $(k-1)$ -type is an element of T .

The set T of the $(k-1)$ -types obtained in this way is finite thanks to Theorem 3.3.1, and corresponds to the k -types of $\mathfrak{A}$ and $\mathfrak{B}$. These k -types are therefore identical. $\square$

Unlike Theorem 3.3.5, the proof of Theorem 3.3.6 does not straightforwardly translate into an algorithm for computing the k -type of the $(0, m)$ -chains described by $S_U^{+\omega}$ from the k -type of the chains in S . The problem is that there are potentially infinitely many sets $S_3^{\tau, j}$ and S_4^j to consider.

Let us show that we can construct the set of $(k-1)$ -types of the chains contained in the sets $S_3^{\tau, j}$ and S_4^j , induced by all possible choices of $j \in \mathbb{N}$, with a finite computation. The main issue is to compute the $(k-1)$ -types of the $(0, m)$ -chains described by

$$S_2 +_U S_2 +_U \cdots +_U S_2, \quad (3.7)$$

where there can be any number of occurrences of $+_U$, from the (supposed unique) $(k-1)$ -type τ_2 of the $(0, m)$ -chains contained in S_2 .

Let $T^0 = \{\tau_2\}$. Then, for each $i \geq 0$, we define

$$T^{i+1} = T^i \cup \{\tau +_U \tau_2 \mid \tau \in T^i\},$$

where $\tau +_U \tau_2$ denotes the $(k-1)$ -type of the $(0, m)$ -chains obtained by applying the operator $+_U$ to a $(0, m)$ -chain of $(k-1)$ -type τ and another of $(k-1)$ -type τ_2 . By Theorem 3.3.5, the resulting $(k-1)$ -type is unique for given k , τ , τ_2 and U . This recurrence relation makes it possible to compute effectively T^i for successive values of i .

By Theorem 3.3.1, the number of possible $(k-1)$ -types for $(0, m)$ -chains is finite for given values of k and m . Since we have $T^i \subseteq T^{i+1}$ for every $i \geq 0$, the computation of T^i will eventually stabilize, i.e., we will have $T^{i+1} = T^i$ for some $i \geq 0$. Such a T^i then contains the $(k-1)$ -types of all the $(0, m)$ -chains of the form (3.7).

Reverse Omega Repetition Let S be a set of $(0, m)$ -chains, with $m \geq 0$. The notation

$$S_U^{-\omega},$$

with $U \subseteq \{1, 2, \dots, m\}$ describes the set of all the $(0, m)$ -chains that can be transformed thanks to Theorem 3.3.2 into chains $(P_1, P_2, \dots, P_m)$ over the domain $(-\infty, 0)$ such that

- for each $j \in \mathbb{N}$, the chain $(P'_1, P'_2, \dots, P'_m)$ belongs to S , where for each $i \in [1, m]$, P'_i is the restriction of P_i to the open interval $(-j-1, -j)$, i.e., $P'_i = P_i \cap (-j-1, -j)$. Like for omega repetition, we transform w.l.o.g. the domain $(-j-1, -j)$ of $(P'_1, P'_2, \dots, P'_m)$ so that it matches the domain of the chains in S ;
- for each $j \in \mathbb{N}_{>0}$ and $i \in \{1, 2, \dots, m\}$, we have $-j \in P_i$ iff $i \in U$.

The operator $\cdot_U^{-\omega}$ produces sets of chains that are the mirror image of those generated by $\cdot_U^{+\omega}$. Here, the domain is split into countably many successive open intervals, with a final one. Inside every such interval, the values of the predicates are described by S . At the junction points between the intervals, the values of the predicates are specified by U .

Theorem 3.3.7. *Let $k \geq 0$. All the chains contained in $S_U^{-\omega}$ share the same k -type, assuming that the same property holds for S .*

Proof. The proof follows the same lines as the one of Theorem 3.3.6. It can be turned in a similar way into an algorithm for computing the k -type of the $(0, m)$ -chains described by $S_U^{-\omega}$ from the k -type of the $(0, m)$ -chains contained in S . $\square$

Shuffle Composition Let $S_1, S_2, \dots, S_\ell$ be sets of $(0, m)$ -chains, with $\ell > 0$ and $m \geq 0$. The notation

$$\llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket, \quad (3.8)$$

where $U_i, U'_i, U''_j \subseteq \{1, 2, \dots, m\}$ for all $i \in [1, \ell]$ and $j \in [1, h]$, with $h > 0$, describes a set S of $(0, m)$ -chains defined as follows. We partition the domain $\mathbb{D}$ of those chains into

- countably infinitely many closed intervals $[a_i, b_i]$, with $a_i, b_i \in \mathbb{D}$ and $a_i < b_i$ for each $i \in \mathbb{N}$. These intervals must be such that any open interval (a, b) , with $a, b \in \mathbb{D}$ and $a < b$, that is not included in any of them intersects infinitely many of them. This implies that, with respect to the natural ordering between the intervals $[a_i, b_i]$, for $i \in \mathbb{N}$, none of them admits an immediate successor or predecessor.
- the remaining points of $\mathbb{D}$, that form a set C that does not include any non-empty open interval. This implies that C is nowhere dense, i.e., that its closure has an empty interior. (Indeed, this closure cannot contain any interior point of an interval $[a_i, b_i]$ of the partition.)

An explicit example of such a partition can be built by a construction similar to that of the Cantor set: Assume w.l.o.g. that $\mathbb{D}$ is the open interval $(0, 1)$. We set $[a_0, b_0] = [\frac{1}{3}, \frac{2}{3}]$, and then carry on the generation of $[a_i, b_i]$ for $i = 1, 2, \dots$ by recursively partitioning the subdomains $(0, \frac{1}{3})$ and $(\frac{2}{3}, 1)$, and so on, following the same procedure.

Once such a partition has been obtained, we assign an index from 1 to ℓ to every interval $[a_i, b_i]$, as well as an index from 1 to h to every element of C . This must be done in such a way that every open interval (a, b) , with $a, b \in \mathbb{D}$ and $a < b$, for which $(a, b) \not\subseteq [a_i, b_i]$ for all $i \in \mathbb{N}$ intersects intervals and remaining points of every index. In other words, every index from 1 to ℓ must be assigned to some $[a_i, b_i]$ such that $[a_i, b_i] \cap (a, b) \neq \emptyset$. Likewise, every index from 1 to h must be assigned to some $c \in C$ such that $a < c < b$.

We are now ready to define the $(0, m)$ -chains that belong to the set S defined by (3.8). These chains are those that can be transformed thanks to Theorem 3.3.2 into chains $(P_1, P_2, \dots, P_m)$ such that:

- for each interval $[a_i, b_i]$ of index j , with $i \in \mathbb{N}$ and $j \in [1, \ell]$:
 - the $(0, m)$ -chain $(P'_1, P'_2, \dots, P'_m)$ belongs to S_j , where each P'_k , with $k \in [1, m]$, is the restriction of the corresponding P_k to the domain (a_i, b_i) , i.e., $P'_k = P_k \cap (a_i, b_i)$.
 - for each $k \in [1, m]$, $a_i \in P_k$ iff $k \in U_j$.
 - for each $k \in [1, m]$, $b_i \in P_k$ iff $k \in U'_j$.
- for each point $c \in C$ of index j , with $j \in [1, h]$, and for each $k \in [1, m]$, $c \in P_k$ iff $k \in U''_j$.

In other words, the $(0, m)$ -chains in S are obtained by densely interleaving closed intervals described by the triplets (U_i, S_i, U'_i) , for $i \in [1, \ell]$. In each such interval, the combinations of values of the predicates at the left and right boundaries are respectively specified by U_i and U'_i , and in the interior of this interval by S_i . At the remaining points of the domain, the predicates admit a dense combination of values specified by the sets U''_i , for $i \in [1, h]$.

Theorem 3.3.8. *Let $k \geq 0$. All the chains contained in*

$$\llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket, \quad (3.9)$$

share the same k -type, assuming that the same property holds for the sets $S_1, S_2, \dots, S_\ell$.

Proof. As for the previous theorems, we prove a stronger property. Let $S_1, S_2, \dots, S_\ell$, with $\ell > 0$, be sets of $(0, m)$ -chains, with $m \geq 0$ that share the same k -type (this type can be different for each S_i). Let S_0 be a set of (n, m) -chains, with $n \geq 0$, that share the same k -type. Let $U_i, U'_i, U''_j \subseteq \{1, 2, \dots, m\}$ for all $i \in [0, \ell]$ and $j \in [1, h]$, with $h > 0$. We will show that the (n, m) -chains that belong to the set described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U_0} S_0 +_{U'_0} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \end{aligned} \quad (3.10)$$

share the same k -type. The statement of the theorem then corresponds to the particular case where $S_0 = S_i$, $U_0 = U_i$ and $U'_0 = U'_i$ for an arbitrary $i \in [1, \ell]$. Indeed, in this case, it follows from the definition of the shuffle composition operator that the sets of $(0, m)$ -chains described by (3.9) and (3.10) are identical. We will rely in the proof on another property of this operator, which is that for any $i \in [1, h]$, the sets of $(0, m)$ -chains described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U''_i} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \end{aligned}$$

and (3.9) are identical as well.

We assume w.l.o.g. that the domains of the chains described by the three successive terms of (3.10) are respectively $(-\infty, -1)$, $(-1, 1)$ and $(1, +\infty)$. In other words, we fix the junction points for the operators $+_{U_0}$ and $+_{U'_0}$ at respectively -1 and 1 .

Let $\mathfrak{A}$ and $\mathfrak{B}$ be two (n, m) -chains belonging to the set described by (3.10). Let us show by induction on k that $\mathfrak{A}$ and $\mathfrak{B}$ have the same k -type.

For the base case $k = 0$, we need to characterize the set of atoms satisfied by $\mathfrak{A}$ and $\mathfrak{B}$. Since these atoms either correspond to $\top$ or involve a first-order variable, they necessarily belong to the 0-type of S_0 . They are thus identical for $\mathfrak{A}$ and $\mathfrak{B}$, hence these two (n, m) -chains have the same 0-type.

For the inductive case $k > 0$, we construct the set T of $(k-1)$ -types of the $(n+1, m)$ -chains obtained by extending $\mathfrak{A}$ and $\mathfrak{B}$ with an additional first-order value $p_{n+1} \in \mathbb{R}$, for all possible choices of this value. The possibilities are the following:

- $p_{n+1} < -1$. Let $\mathfrak{A} = (p_1, p_2, \dots, p_n, P_1, P_2, \dots, P_m)$. By definition of the shuffle composition operator, there exists a partition Π of $(-\infty, -1)$ into

closed intervals and remaining points, over which the values of the predicates P_j within $(-\infty, -1)$ are characterized by the triplets (U_1, S_1, U'_1) , $(U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell)$ and the sets $U''_1, U''_2, \dots, U''_h$, for all $j \in [1, m]$. We denote those closed intervals by $[a_i, b_i]$, for $i \in \mathbb{N}$, and the set of remaining points by C . There are several possibilities:

- $a_i < p_{n+1} < b_i$ for some interval $[a_i, b_i]$ in Π , with $i \in \mathbb{N}$. Let $\eta \in [1, \ell]$ be the index assigned to this interval. This means that the restriction of $(P_1, P_2, \dots, P_m)$ to the interval (a_i, b_i) belongs to S_η , assuming (w.l.o.g. thanks to Theorem 3.3.2) that the domain of S_η is (a_i, b_i) . The k -type of the chains in S_η is unique by hypothesis. Each element τ of this k -type provides the $(k-1)$ -type of some set S_η^τ of $(1, m)$ -chains obtained by adding a first-order value p_{n+1} to a $(0, m)$ -chain in S_η . For each S_η^τ obtained in this way, the $(n+1, m)$ -chains described by

$$R_\eta^\tau = S_\eta^\tau +_{U'_\eta} \llbracket (U_1, S_1, U'_1), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, \dots, U''_h \rrbracket +_{U_0} S_0$$

share the same $(k-1)$ -type by inductive hypothesis and by Theorem 3.3.5. Therefore, by inductive hypothesis, the $(n+1, m)$ -chains described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U'_\eta} R_\eta^\tau +_{U'_0} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \end{aligned}$$

share the same $(k-1)$ -type. After applying the reordering operator with respect to the permutation $[2; 3; \dots; n+1; 1]$, one obtains a $(k-1)$ -type that is an element of T .

- $p_{n+1} = a_i$ for some interval $[a_i, b_i]$ in Π , with $i \in \mathbb{N}$. Let $\eta \in [1, \ell]$ be the index assigned to this interval. Up to some reordering of the variables, the $(n+1, m)$ -chain obtained by extending $\mathfrak{A}$ with p_{n+1} belongs to the set described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U''_1} R_\eta +_{U'_0} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket, \end{aligned} \tag{3.11}$$

where R_η is the set of $(n+1, m)$ -chains described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U'_\eta}^1 S_\eta +_{U'_\eta} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U_0} S_0. \end{aligned}$$

By inductive hypothesis and Theorem 3.3.5, the $(n+1, m)$ -chains described by (3.11) share the same $(k-1)$ -type. After applying the reordering operator with respect to the permutation $[2; 3; \dots; n+1; 1]$, one obtains a $(k-1)$ -type that is an element of T .

- $p_{n+1} = b_i$ for some interval $[a_i, b_i]$ in Π , with $i \in \mathbb{N}$. Let $\eta \in [1, \ell]$ be the index assigned to this interval. Up to some reordering of the variables, the $(n+1, m)$ -chain obtained by extending $\mathfrak{A}$ with p_{n+1} belongs to the set described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U_\eta} R'_\eta +_{U'_0} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket, \end{aligned} \quad (3.12)$$

where R'_η is the set of $(n+1, m)$ -chains described by

$$S_\eta +_{U'_\eta} \llbracket (U_1, S_1, U'_1), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, \dots, U''_h \rrbracket +_{U_0} S_0.$$

By inductive hypothesis and Theorem 3.3.5, the $(n+1, m)$ -chains described by (3.12) share the same $(k-1)$ -type. After applying the reordering operator with respect to the permutation $[2; 3; \dots; n+1; 1]$, one obtains a $(k-1)$ -type that is an element of T .

- $p_{n+1} \in C$. Let $\eta \in [1, h]$ be the index assigned to p_{n+1} . Up to some reordering of the variables, the $(n+1, m)$ -chain obtained by extending $\mathfrak{A}$ with p_{n+1} belongs to the set described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U''_1} R''_\eta +_{U'_0} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket, \end{aligned} \quad (3.13)$$

where R''_η is the set of $(n+1, m)$ -chains described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U''_\eta} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U_0} S_0. \end{aligned}$$

By inductive hypothesis and Theorem 3.3.5, the $(n+1, m)$ -chains described by (3.13) share the same $(k-1)$ -type. After applying the reordering operator with respect to the permutation $[2; 3; \dots; n+1; 1]$, one obtains a $(k-1)$ -type that is an element of T .

- $p_{n+1} > 1$. This is the symmetrical situation of the previous one. It can be handled in a similar way.
- $-1 < p_{n+1} < 1$. Then, each element τ of the k -type of the chains in S_0 (supposed to be unique by hypothesis) provides the $(k-1)$ -type of some set S_0^τ of $(n+1, m)$ -chains obtained by adding such a first-order value p_{n+1} to a (n, m) -chain in S_0 . For each S_0^τ obtained in this way, the $(n+1, m)$ -chains described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U_0} S_0^\tau +_{U'_0} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \end{aligned}$$

share the same $(k-1)$ -type by inductive hypothesis. This $(k-1)$ -type is an element of T .

- $p_{n+1} = -1$. Then, up to some reordering of the variables, the $(n+1, m)$ -chain obtained by extending $\mathfrak{A}$ with p_{n+1} belongs to the set described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U'_1} R +_{U'_0} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket, \end{aligned} \quad (3.14)$$

where R is the set of $(n+1, m)$ -chains described by

$$\llbracket (U_1, S_1, U'_1), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, \dots, U''_h \rrbracket +_{U_0} S_0.$$

By inductive hypothesis and Theorem 3.3.5, the $(n+1, m)$ -chains described by (3.14) share the same $(k-1)$ -type. After applying the reordering operator with respect to the permutation $[2; 3; \dots; n+1; 1]$, one obtains a $(k-1)$ -type that is an element of T .

- $p_{n+1} = 1$. Then, the $(n+1, m)$ -chain obtained by extending $\mathfrak{A}$ with p_{n+1} belongs to the set described by

$$\begin{aligned} & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket \\ & \quad +_{U_0} R' +_{U'_1} \\ & \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket, \end{aligned} \quad (3.15)$$

where R' is the set of $(n+1, m)$ -chains described by

$$S_0 +_{U_0} \llbracket (U_1, S_1, U'_1), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, \dots, U''_h \rrbracket.$$

By inductive hypothesis and Theorem 3.3.5, the $(n+1, m)$ -chains described by (3.15) share the same $(k-1)$ -type. This $(k-1)$ -type is an element of T .

The set T of the $(k-1)$ -types obtained in this way does only depend on the choice of the sets S_i and U_i, U'_i, U''_j , for $i \in [0, \ell]$ and $j \in [0, h]$, and not on the particular values of p_i and P_j , for $p_i \in [1, n]$ and $P_j \in [1, m]$. This set T is thus identical for $\mathfrak{A}$ and $\mathfrak{B}$, which therefore have the same k -type. $\square$

Note that the proof of Theorem 3.3.8 provides an algorithm for computing the k -type of the chains described by

$$\llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket$$

from the k -types of the sets $S_1, S_2, \dots, S_\ell$, for any $k \geq 0$.

Synthesis

The following definition characterizes the class of chains that can be constructed by starting from the elementary chains of Section 3.3.3, and applying finitely many times the operators of Section 3.3.3.

Definition 3.3.1. A (n, m) -chain, with $n, m \geq 0$, is said to be constructible if it belongs to a set of chains described by

- $\llbracket U_1, U_2, \dots, U_\ell \rrbracket$ for some $\ell > 0$ and $U_i \subseteq \{1, 2, \dots, m\}$ for all $i \in [1, \ell]$, with $n = 0$,
- S/V , where V is a permutation of $[1; 2; \dots; n]$ and S is a set of constructible (n, m) -chains,
- $S_1 +_U^\ell S_2$, where $0 \leq \ell \leq n$, $U \subseteq \{1, 2, \dots, m\}$, S_1 is a set of constructible (n_1, m) -chains, and S_2 is a set of constructible (n_2, m) -chains, with $n_1 + \ell + n_2 = n$,
- $S_U^{+\omega}$ or $S_U^{-\omega}$, where $n = 0$, $U \subseteq \{1, 2, \dots, m\}$ and S is a set of constructible $(0, m)$ -chains, or
- $\llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket$, where $n = 0$, $\ell, h > 0$, $U_i, U'_i, U''_j \subseteq \{1, 2, \dots, m\}$ for all $i \in [1, \ell]$ and $j \in [1, h]$, and $S_1, S_2, \dots, S_\ell$ are sets of constructible $(0, m)$ -chains.

Definition 3.3.1 provides a symbolic notation for sets of chains: we call a *chain term* an expression obtained by applying finitely many times reordering, concatenation, repetition or shuffle composition operators to elementary sets of chains of the form $\llbracket U_1, U_2, \dots, U_\ell \rrbracket$. In Sections 3.3.3 and 3.3.3, we have established that for any $n, m, k \geq 0$, the (n, m) -chains described by a chain term share the same k -type. Moreover, we have developed algorithms for computing the k -type of the chains in every elementary set, as well as in every set obtained by applying reordering, concatenation, repetition or shuffle composition operators to sets of chains with a known k -type. This leads to the following result.

Theorem 3.3.9. *There exists an algorithm for computing, for any $k, n, m \geq 0$, the k -type of the constructible (n, m) -chains described by a given chain term.*

3.3.4 Deciding Satisfiability

By applying the results of Section 3.3.2, one obtains the following corollary to Theorem 3.3.9.

Corollary 3.3.2. *For every constructible (n, m) -chain $\mathfrak{A}$, with $n, m \geq 0$ described by a chain term and formula φ of $\text{MFO}(\mathbb{R}, <)$, it is decidable whether $\mathfrak{A} \models \varphi$.*

Our goal now consists in establishing that for every (n, m) -chain $\mathfrak{A}$, with $n, m \geq 0$, and $k \geq 0$, there exists a constructible (n, m) -chain $\mathfrak{B}$ such that $\mathfrak{A} \equiv_k \mathfrak{B}$. An important consequence of this property will be that every formula of $\text{MFO}(\mathbb{R}, <)$ that is satisfiable admits a model that takes the form of a constructible chain.

The results of this section follow the same approach as in [73], working in $\mathbb{R}$ instead of in an arbitrary totally ordered set.

Good Chains

A (n, m) -chain $\mathfrak{A}$, with $n, m \geq 0$, is said to be *k-good*, with $k \geq 0$, if there exists a constructible (n, m) -chain $\mathfrak{B}$ such that $\mathfrak{A} \equiv_k \mathfrak{B}$. From Definition 3.3.1 and the results of Section 3.3.3, we have the following lemma (corresponding to Lemma 7 of [73]), stating that the class of *k-good* chains is closed under reordering, concatenation, omega and reverse omega repetition, and shuffle composition operators.

Lemma 3.3.1. *The following properties hold:*

- If S is a set of *k-good* (n, m) -chains for $k, n, m \geq 0$, then S/V is a set of *k-good* (n, m) -chains, for any permutation V of $[1; 2; \dots; n]$.
- If S_1 and S_2 are sets of *k-good* (n_1, m) and (n_2, m) -chains respectively, for $k, n_1, n_2, m \geq 0$, then $S_1 +_U^j S_2$ is a set of *k-good* $(n_1 + j + n_2, m)$ -chains, for any $j \geq 0$ and $U \subseteq \{1, 2, \dots, m\}$.
- If $S_1, S_2, \dots, S_\ell$ are sets of *k-good* $(0, m)$ -chains, for $k, m \geq 0$ and $\ell \geq 1$, then the $(0, m)$ -chains in $\llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket$ are *k-good*, for any $U_i, U'_i, U''_j \subseteq \{1, 2, \dots, m\}$, for all $i \in [1, \ell]$, $j \in [1, h]$, with $h > 0$.
- If S is a set of *k-good* $(0, m)$ -chains for $k, m \geq 0$, then $(S)_U^{+\omega}$ and $(S)_U^{-\omega}$ are sets of *k-good* $(0, m)$ -chains, for any $U \subseteq \{1, 2, \dots, m\}$.

Segments

In the following definition, we abuse the notation by writing $\mathfrak{A}$ for the set $\{\mathfrak{A}\}$ containing only the chain $\mathfrak{A}$, for the sake of simplicity.

Definition 3.3.2. A (n_1, m) -chain $\mathfrak{A}_1$ is a segment of a (n_2, m) -chain $\mathfrak{A}_2$, with $0 \leq n_1 \leq n_2$ and $m \geq 0$, if at least one of the following conditions is satisfied:

- There exists a permutation V of $[1; 2; \dots; n_2]$ such that

$$\mathfrak{A}_2 = \mathfrak{A}_1/V. \quad (3.16)$$

- There exist $n_3 \geq 0$, a (n_3, m) -chain $\mathfrak{A}_3$, a set $U \subseteq \{1, 2, \dots, m\}$, $\ell \geq 0$, and a permutation V of $[1; 2; \dots; n_2]$ such that $n_2 = n_3 + \ell + n_1$ and

$$\mathfrak{A}_2 = (\mathfrak{A}_3 +_U^\ell \mathfrak{A}_1)/V. \quad (3.17)$$

- There exist $n_4 \geq 0$, a (n_4, m) -chain $\mathfrak{A}_4$, a set $U \subseteq \{1, 2, \dots, m\}$, $\ell \geq 0$, and a permutation V of $[1; 2; \dots; n_2]$ such that $n_2 = n_1 + \ell + n_4$ and

$$\mathfrak{A}_2 = (\mathfrak{A}_1 +_U^\ell \mathfrak{A}_4)/V. \quad (3.18)$$

- There exist $n_3, n_4 \geq 0$, a (n_3, m) -chain $\mathfrak{A}_3$, a (n_4, m) -chain $\mathfrak{A}_4$, sets $U_1, U_2 \subseteq \{1, 2, \dots, m\}$, $\ell_1, \ell_2 \geq 0$, and a permutation V of $[1; 2; \dots; n_2]$ such that $n_2 = n_3 + \ell_1 + n_1 + \ell_2 + n_4$ and

$$\mathfrak{A}_2 = (\mathfrak{A}_3 +_{U_1}^{\ell_1} \mathfrak{A}_1 +_{U_2}^{\ell_2} \mathfrak{A}_4)/V. \quad (3.19)$$

In this last case, the segment $\mathfrak{A}_1$ is said to be bounded.

The bounded segment $\mathfrak{A}_1$ describes the content of the chain $\mathfrak{A}_2$ between two points a and b in $\mathbb{R}$, which correspond to the junction points of the two concatenation operators in (3.19). We will write

$$\mathfrak{A}_1 = [\mathfrak{A}_2]_{(a,b)}$$

when we want to make this property explicit, in the particular case that the first-order variables reordering is irrelevant, that is, (3.19) holds with either $n_1 = 0$, or $n_1 = n_2 > 0$ and $V = [1; 2; \dots; n_2]$. Similarly, for unbounded segments, we write $\mathfrak{A}_1 = [\mathfrak{A}_2]_{(-\infty, +\infty)}$, $\mathfrak{A}_1 = [\mathfrak{A}_2]_{(-\infty, b)}$ and $\mathfrak{A}_1 = [\mathfrak{A}_2]_{(a, +\infty)}$ when (respectively) (3.16), (3.18) and (3.17) are satisfied, under the same assumption on n_1 , n_2 and V .

The following result corresponds to Lemma 8 of [73].

Lemma 3.3.2. *If every bounded segment of a (n, m) -chain $\mathfrak{A}$ is k -good, for $n, m, k \geq 0$, then $\mathfrak{A}$ is k -good as well.*

Proof. It is always possible to decompose $\mathfrak{A}$ into

$$\mathfrak{A} = (\mathfrak{A}_1 +_U \mathfrak{A}_2)/V \quad (3.20)$$

where $\mathfrak{A}_1$ and $\mathfrak{A}_2$ are respectively (n_1, m) and (n_2, m) -chains, with $n_1 + n_2 = n$, $U \subseteq \{1, 2, \dots, m\}$, and V is a permutation of $[1; 2; \dots; n_1 + n_2]$. Indeed, it suffices to fix the junction point of the concatenation operator at any point $p \in \mathbb{R}$ that does not coincide with a first-order value in $\mathfrak{A}$. Let us consider (w.l.o.g. thanks to Theorem 3.3.2) that we have $p = 0$. The second-order values that are true at 0 then determine the set U .

By Lemma 3.3.1, to prove that $\mathfrak{A}$ is k -good, it is sufficient to establish that $\mathfrak{A}_1$ and $\mathfrak{A}_2$ are both k -good. We only give the proof for $\mathfrak{A}_2$; the case of $\mathfrak{A}_1$ is symmetric and can be handled in a similar way.

Let $\mathfrak{A} = (p_1, p_2, \dots, p_n, P_1, P_2, \dots, P_m)$. We pick an arbitrary infinite sequence $d_0, d_1, d_2, \dots \in \mathbb{R}_{>0}$ of points in the domain $(0, +\infty)$ of $\mathfrak{A}_2$, such that

- $d_{i+1} > d_i$ for all $i \in \mathbb{N}$. In other words, the sequence is strictly increasing.
- $\forall d > 0 : \exists i \in \mathbb{N} : d_i > d$. In other words, the sequence does not admit an upper bound in $\mathbb{R}$.

- $p_i < d_0$ for all $i \in [1, n]$. In other words, the sequence begins after the greatest first-order value in $\mathfrak{A}$.

Every unordered pair of values (d_i, d_j) with $i, j \in \mathbb{N}$ and $i \neq j$ determines a bounded segment of $\mathfrak{A}$, which corresponds to $[\mathfrak{A}]_{(d_i, d_j)}$ if $d_i < d_j$ and to $[\mathfrak{A}]_{(d_j, d_i)}$ if $d_i > d_j$. We can partition the set of all pairs (d_i, d_j) according to

- the k -type of the bounded segment that they determine, together with
- the combination of values taken by the predicates $P_1, P_2, \dots, P_m$ at $d_{i,j} = \min(d_i, d_j)$. Formally, this combination is represented by the set $U_{i,j} = \{\ell \in [1, m] \mid d_{i,j} \in P_\ell\}$.

The bounded segments involved in this partitioning scheme have the dimension $(0, m)$, hence by Theorem 3.3.1, their k -type can only take finitely many values. The possible values of the sets $U_{i,j}$ are also finite. It follows that our partition is finite.

By the infinite version of Ramsey's theorem [92], there exists an infinite subsequence $\delta_0, \delta_1, \delta_2, \dots$ of $d_0, d_1, d_2, \dots$ such that every pair (δ_i, δ_j) with $i, j \in \mathbb{N}$ and $i \neq j$ belongs to the same class of the partition. Note that this sequence $\delta_0, \delta_1, \delta_2, \dots$ does not admit an upper bound in $\mathbb{R}$. It is such that the k -type of the bounded interval $[\mathcal{A}]_{(\delta_i, \delta_{i+1})}$ is identical for all $i \in \mathbb{N}$, and the combination of values taken by the predicates $P_1, P_2, \dots, P_m$ at all the δ_i is identical as well. We therefore have

$$\mathfrak{A}_2 \equiv_k [\mathfrak{A}]_{(0, \delta_0)} +_{U'} ([\mathfrak{A}]_{(\delta_0, \delta_1)})_{U'}^{+\omega},$$

for some $U' \subseteq \{1, 2, \dots, m\}$.

By hypothesis, the bounded segments $[\mathfrak{A}]_{(0, \delta_0)}$ and $[\mathfrak{A}]_{(\delta_0, \delta_1)}$ involved in this expression are k -good. It follows by Lemma 3.3.1 that $\mathfrak{A}_2$ is k -good as well. $\square$

The reciprocal of the property expressed by Lemma 3.3.2 also holds, for bounded and as well as unbounded segments:

Lemma 3.3.3. *Every segment of a k -good (n, m) -chain $\mathfrak{A}$ is k -good as well, for any $n, m, k \geq 0$.*

Proof. The proof is by structural induction on the description of a set of chains that are k -elementarily equivalent to $\mathfrak{A}$. For the base case where $\mathfrak{A} \in \llbracket U_1, U_2, \dots, U_\ell \rrbracket$ for some $U_1, U_2, \dots, U_\ell \subseteq \{1, 2, \dots, m\}$, with $\ell > 0$, we have that every segment of $\mathfrak{A}$ also belongs to $\llbracket U_1, U_2, \dots, U_\ell \rrbracket$ by definition of shuffle chains. Every such segment is thus k -good.

For the inductive case, we consider separately each operator.

- Every segment $\mathfrak{A}$ of a (n, m) -chain in S/V , where S is a set of k -good chains and V is a permutation of $[1; 2; \dots; n]$, is such that

$$\mathfrak{A} = \mathfrak{A}_1/V_1,$$

where the (n_1, m) -chain $\mathfrak{A}_1$ is a segment of a chain in S , with $0 \leq n_1 \leq n$, and V_1 is a permutation of $[1; 2; \dots; n_1]$. By inductive hypothesis and Lemma 3.3.1, $\mathfrak{A}$ is k -good.

- Every segment $\mathfrak{A}$ of a (n, m) -chain in $S_1 +_U^\ell S_2$, where S_1, S_2 are sets of k -good chains, $\ell \geq 0$ and $U \subseteq \{1, 2, \dots, m\}$, is such that either
 - $\mathfrak{A}$ is a segment of a chain belonging to S_1 or S_2 . In this case, $\mathfrak{A}$ is k -good by inductive hypothesis,
 - or $\mathfrak{A} = (\mathfrak{A}_1 +_U^\ell \mathfrak{A}_2)/V$, where $\mathfrak{A}_1$ is a segment of dimension (n_1, m) of a chain in S_1 , with $n_1 \geq 0$, $\mathfrak{A}_2$ is a segment of dimension (n_2, m) of a chain in S_2 , with $n_2 \geq 0$, and V is a permutation of $[1; 2; \dots; n_1 + n_2 + \ell]$. This implies, by inductive hypothesis and Lemma 3.3.1, that $\mathfrak{A}$ is k -good.
- Every segment $\mathfrak{A}$ of a $(0, m)$ -chain in $S_U^{+\omega}$ or $S_U^{-\omega}$, where S is a set of k -good $(0, m)$ -chains and $U \subseteq \{1, 2, \dots, m\}$, is such that
 - either $\mathfrak{A}$ is a segment of a chain belonging to S . In this case, $\mathfrak{A}$ is k -good by inductive hypothesis.
 - or $\mathfrak{A} = \mathfrak{A}_1 +_U \mathfrak{A}_2 +_U \mathfrak{A}_3 +_U \dots +_U \mathfrak{A}_\ell +_U \mathfrak{A}_{\ell+1}$, where there are any (nonzero) number of occurrences of the operator $+_U$, $\mathfrak{A}_1$ and $\mathfrak{A}_{\ell+1}$ are segments of $(0, m)$ -chains in S , and $\mathfrak{A}_2$ is a $(0, m)$ -chain in S . By inductive hypothesis and Lemma 3.3.1, $\mathfrak{A}$ is then k -good.
- Every segment $\mathfrak{A}$ of a $(0, m)$ -chain in

$$[(U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h],$$

where $\ell, h \geq 0$, $S_1, S_2, \dots, S_\ell$ are sets of k -good $(0, m)$ -chains and $U_i, U'_i, U''_j \subseteq \{1, 2, \dots, m\}$ for all $i \in [0, \ell]$ and $j \in [0, h]$, is such that

- either $\mathfrak{A}$ is a segment of a chain belonging to $S_1, S_2, \dots$ or S_ℓ . In this case, $\mathfrak{A}$ is k -good by inductive hypothesis.
- $\mathfrak{A} = \mathfrak{A}_1 +_{U'_i} \mathfrak{A}_2 +_{U_j} \mathfrak{A}_3$, where $\mathfrak{A}_1$ is a segment of a $(0, m)$ -chain in some set S_i , with $i \in [1, \ell]$, $\mathfrak{A}_3$ is a segment of a $(0, m)$ -chain in some set S_j , with $j \in [1, \ell]$, and $\mathfrak{A}_2$ is a $(0, m)$ -chain in

$$[(U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h].$$

By inductive hypothesis and Lemma 3.3.1, this implies that $\mathfrak{A}$ is k -good.

- $\mathfrak{A} = \mathfrak{A}_1 +_{U'_i} \mathfrak{A}_2$, where $\mathfrak{A}_1$ is a segment of a $(0, m)$ -chain in some set S_i , with $i \in [1, \ell]$ and $\mathfrak{A}_2$ is a $(0, m)$ -chain in

$$[(U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h].$$

By inductive hypothesis and Lemma 3.3.1, this implies that $\mathfrak{A}$ is k -good.

- or $\mathfrak{A} = \mathfrak{A}_2 +_{U_j} \mathfrak{A}_3$, where $\mathfrak{A}_3$ is a segment of a $(0, m)$ -chain in some set S_j , with $j \in [1, \ell]$, and $\mathfrak{A}_2$ is a $(0, m)$ -chain in

$$\llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket.$$

By inductive hypothesis and Lemma 3.3.1, this implies that $\mathfrak{A}$ is k -good.

□

Lemmas 3.3.2 and 3.3.3 reduce the problem of establishing that a chain is k -good with respect to its full domain to the same problem over bounded subsets of this domain. This result is quite powerful. Consider for instance a $(0, m)$ -chain $\mathfrak{A}$ defined over the domain $(0, +\infty)$, such that for every $i \in \mathbb{N}$, the bounded segment $[\mathfrak{A}]_{(i, i+1)}$ belongs to the set described by $(\dots((S_U^{+\omega})_U^{+\omega})_U^{+\omega} \dots)^{+\omega}$, where there are i occurrences of the repetition operator $\cdot_U^{+\omega}$, S is a set of k -good $(0, m)$ -chains, and $U \subseteq \{1, 2, \dots, m\}$. In other words, we have (using improperly the notation):

$$\mathfrak{A} = \mathfrak{A}_0 +_U (\mathfrak{A}_1)^{+\omega} +_U ((\mathfrak{A}_2)^{+\omega})^{+\omega} +_U (((\mathfrak{A}_3)^{+\omega})^{+\omega})^{+\omega} +_U \dots,$$

where $\mathfrak{A}_i$ is k -good for every $i \in \mathbb{N}$. Even though the previous expression is not a valid chain term — since it involves infinitely many occurrences of the concatenation operator $+_U$ — it follows from Lemmas 3.3.2 and 3.3.3 that the chain $\mathfrak{A}$ is k -good, and therefore that it belongs to some set that can be described by a chain term.

Goodness Is Everywhere

The following result corresponds to Lemma 9 of [73], adapted to the domain $\mathbb{R}$.

Lemma 3.3.4. *For every $k \geq 0$ and (n, m) -chain $\mathfrak{A}$, with $n, m \geq 0$, $\mathfrak{A}$ is k -good.*

Proof. We first observe that if $n > 0$, then $\mathfrak{A}$ can be decomposed into

$$\mathfrak{A} = (\mathfrak{A}_1 +_{U_1}^{\ell_1} \mathfrak{A}_2 +_{U_2}^{\ell_2} \dots +_{U_j}^{\ell_j} \mathfrak{A}_{j+1})/V,$$

where $0 < j \leq n$, $\ell_1, \ell_2, \dots, \ell_j \geq 0$ with $\ell_1 + \ell_2 + \dots + \ell_j = n$, $U_i \subseteq \{1, 2, \dots, m\}$ for all $i \in [1, j]$, V is a permutation of $[1, 2, \dots, n]$, and $\mathfrak{A}_i$ is a $(0, m)$ -chain for each $i \in [1, j+1]$. By Lemma 3.3.1, it is then sufficient to establish the lemma for each $\mathfrak{A}_i$, i.e., we can assume w.l.o.g. that $n = 0$.

Let therefore $\mathfrak{A} = (P_1, P_2, \dots, P_m)$ be a $(0, m)$ -chain. Recall that the domain is assumed to be $\mathbb{R}$. We define the set $R \subseteq \mathbb{R}$ as the union of the intervals (a, b) , with $a \in \mathbb{R} \cup \{-\infty\}$, $b \in \mathbb{R} \cup \{+\infty\}$ and $a < b$, for which there exist $\ell > 0$ and $U_1, U_2, \dots, U_\ell \subseteq \{1, 2, \dots, m\}$ such that $[\mathfrak{A}]_{(a, b)} \in \llbracket U_1, U_2, \dots, U_\ell \rrbracket$.

Let us show that the set R is dense in $\mathbb{R}$, i.e., that every $a, b \in \mathbb{R}$ with $a < b$ are such that $R \cap (a, b) \neq \emptyset$. By contradiction, assume that there exists an

non-empty open interval (a, b) of $\mathbb{R}$ inside which no bounded segment of $\mathfrak{A}$ is a constant or shuffle chain. We define

$$\mathcal{U}_{(a,b)} = \{\{j \in [1, m] \mid p \in P_j\} \mid p \in \mathbb{R} \wedge a < p < b\}.$$

In other words, $\mathcal{U}_{(a,b)}$ describes the combinations of predicates values that occur inside the interval (a, b) .

We then perform the following operation. If there exist $\alpha, \beta \in \mathbb{R}$ such that $a < \alpha < \beta < b$, i.e., (α, β) is a proper sub-interval of (a, b) , and $\mathcal{U}_{(\alpha,\beta)} \subsetneq \mathcal{U}_{(a,b)}$, then we replace (a, b) with (α, β) and repeat the same procedure. Note that, since m is fixed, the number of possible distinct sets $\mathcal{U}_{(a,b)}$ is finite, hence this procedure will terminate after a finite number of steps. The final interval (a, b) is then such that $[A]_{(a,b)} \in \llbracket U_1, U_2, \dots, U_\ell \rrbracket$, where $U_1, U_2, \dots, U_\ell$ are the elements of $\mathcal{U}_{(a,b)}$. This contradicts the hypothesis that within our original interval (a, b) , the chain $\mathfrak{A}$ does not admit a constant or shuffle bounded segment.

At this stage, we have established that there exists a partition Π_1 of the domain $\mathbb{R}$ into

- open intervals (a_i, b_i) , with $a_i \in \mathbb{R} \cup \{-\infty\}$, $b_i \in \mathbb{R} \cup \{+\infty\}$ and $a_i < b_i$ for either all $i \in \mathbb{N}$ or all $i \in [1, q]$ with $q > 0$. For each such interval, there exist $\ell_i > 0$ and $U_{i,1}, U_{i,2}, \dots, U_{i,\ell_i} \subseteq \{1, 2, \dots, m\}$ such that $[\mathfrak{A}]_{(a_i, b_i)} \in \llbracket U_{i,1}, U_{i,2}, \dots, U_{i,\ell_i} \rrbracket$. Note that the number of such intervals is necessarily finite or countable, since every of them contains a rational value that does not belong to the others. The set of all those intervals is denoted by I_1 .
- remaining points $c \in \mathbb{R}$. The set C_1 of those remaining points does not include any non-empty open interval, i.e., C_1 is nowhere dense. (Note that C_1 is the complement of a union of open sets, and is therefore closed.)

We now define a relation $\sim$ on the elements of I_1 : Two intervals $(a_1, b_1), (a_2, b_2) \in I_1$ are such that $(a_1, b_1) \sim (a_2, b_2)$ iff either

- $(a_1, b_1) = (a_2, b_2)$,
- $a_1 < b_2$ and the segment $[\mathfrak{A}]_{(a_1, b_2)}$ is k -good, or
- $a_2 < b_1$ and the segment $[\mathfrak{A}]_{(a_2, b_1)}$ is k -good.

Let us show that $\sim$ is an equivalence relation. Indeed, if $(a_1, b_1) \sim (a_2, b_2)$ and $(a_2, b_2) \sim (a_3, b_3)$, then one has $(a_1, b_1) \sim (a_3, b_3)$ as a consequence of the definition of $\sim$ together with Lemmas 3.3.1 and 3.3.3.

Each equivalence class E of $\sim$ is a set of disjoint open intervals (a_i, b_i) . For each such class, we define

$$\alpha_E = \inf\{a_i \mid (a_i, b_i) \in E\},$$

and

$$\beta_E = \sup\{b_i \mid (a_i, b_i) \in E\}.$$

We thus have $\alpha_E \in \mathbb{R} \cup \{-\infty\}$, $\beta_E \in \mathbb{R} \cup \{+\infty\}$ and $\alpha_E < \beta_E$.

Each open interval (α_E, β_E) is such that the segment $[\mathfrak{A}]_{(\alpha_E, \beta_E)}$ is k -good. Indeed, for every bounded sub-interval (a, b) of (α_E, β_E) , i.e., such that $\alpha_E < a < b < \beta_E$, there exist $(a_1, b_1) \in E$ and $(a_2, b_2) \in E$ such that $a_1 < a$ and $b < b_2$. By definition of the equivalence relation $\sim$, this implies that $[\mathcal{A}]_{(a_1, b_2)}$ is k -good, from which Lemma 3.3.3 entails that $[\mathfrak{A}]_{(a, b)}$ is k -good. Since this property holds for every bounded segment of $[\mathfrak{A}]_{(\alpha_E, \beta_E)}$, it follows from Lemma 3.3.2 that $[\mathcal{A}]_{(\alpha_E, \beta_E)}$ is k -good.

By merging together equivalent intervals belonging to I_1 , i.e., by computing the new set of intervals $I_2 = I_1 / \sim$ where $/$ denotes partition refinement, and removing from C_1 the points that are covered by intervals in I_2 , i.e., by setting $C_2 = C_1 \setminus \cup_{(a_i, b_i) \in I_2} (a_i, b_i)$, we obtain a partition Π_2 of $\mathbb{R}$ into

- a set I_2 of open intervals (a_i, b_i) , with $a_i \in \mathbb{R} \cup \{-\infty\}$, $b_i \in \mathbb{R} \cup \{+\infty\}$ and $a_i < b_i$ for either all $i \in \mathbb{N}$ or all $i \in [1, q]$ with $q > 0$. For each such interval, the segment $[\mathfrak{A}]_{(a_i, b_i)}$ is k -good.
- a closed set C_2 of remaining points. This set does not include any non-empty open interval, i.e., C_2 is nowhere dense.

The intervals in I_2 cannot be adjacent, i.e., I_2 does not contain intervals (a_1, b_1) and (a_2, b_2) such that $b_1 = a_2$. Indeed, in such a case, the segment $[\mathfrak{A}]_{(a_1, b_2)}$ would be k -good thanks to Lemma 3.3.1, and the interval (a_1, b_2) would belong to I_2 . (Recall that the intervals in I_2 are elements of a partition of $\mathbb{R}$, hence they are disjoint.)

We conclude the proof by showing that I_2 can only contain a single interval $(-\infty, +\infty)$, which implies that $\mathfrak{A}$ is k -good. Assume, by contradiction, that I_2 contains at least two distinct intervals (a_1, b_1) and (a_2, b_2) . We can impose w.l.o.g. $b_1 < a_2$. For every such pair of intervals, we define

- the set $\mathcal{T}_{(a_1, b_1), (a_2, b_2)}$ of k -types of all the intervals $(\alpha, \beta) \in I_2$ such that $b_1 < \alpha < \beta < a_2$. This set cannot be empty, otherwise C_2 would include a non-empty open interval within (b_1, a_2) .
- the set $\mathcal{U}_{(a_1, b_1), (a_2, b_2)} = \{\{j \in [1, m] \mid p \in P_j\} \mid p \in C_2 \wedge b_1 < p < a_2\}$, that characterizes the combinations of values of the predicates that occur between b_1 and a_2 . This set cannot be empty, since $C_2 \cap (b_1, a_2) \neq \emptyset$. (Otherwise, (b_1, a_2) would belong to I_2 , which is impossible since this interval is adjacent to both (a_1, b_1) and (a_2, b_2) .)

We then perform the following operation. If there exist $(a_3, b_3) \in I_2$ and $(a_4, b_4) \in I_2$ such that $b_1 < a_3 < b_3 < a_4 < b_4 < a_2$, and

$$\mathcal{T}_{(a_3, b_3), (a_4, b_4)} \subsetneq \mathcal{T}_{(a_1, b_1), (a_2, b_2)} \text{ or } \mathcal{U}_{(a_3, b_3), (a_4, b_4)} \subsetneq \mathcal{U}_{(a_1, b_1), (a_2, b_2)},$$

we replace (a_1, b_1) by (a_3, b_3) and (a_2, b_2) by (a_4, b_4) , and repeat the same procedure. Note that, by Theorem 3.3.1, the set of possible k -types for the segments $[\mathfrak{A}]_{(a_i, b_i)}$ with $(a_i, b_i) \in I_2$ is finite, hence the number of possible sets $\mathcal{T}_{(a_1, b_1), (a_2, b_2)}$ is finite. Furthermore, the number of possible sets $\mathcal{U}_{(a_1, b_1), (a_2, b_2)}$

is finite as well since m is fixed. This procedure will thus terminate in a finite number of steps. The final set $\mathcal{T}_{(a_1, b_1), (a_2, b_2)}$ will be non-empty, and will contain the k -types of some number $\ell > 0$ of sets $S_1, S_2, \dots, S_\ell$ of k -good $(0, m)$ -chains, corresponding to bounded segments $[\mathfrak{A}]_{(\alpha, \beta)}$ for some intervals $(\alpha, \beta) \in I_2$. The final set $\mathcal{U}_{(a_1, b_1), (a_2, b_2)}$ will also be non-empty, and will contain some number $h > 0$ of sets $U_1, U_2, \dots, U_h \subseteq [1, m]$. One then has

$$[\mathfrak{A}]_{(b_1, a_2)} \in \llbracket (U_1, S_1, U'_1), (U_2, S_2, U'_2), \dots, (U_\ell, S_\ell, U'_\ell), U''_1, U''_2, \dots, U''_h \rrbracket.$$

This implies that $[\mathfrak{A}]_{(b_1, a_2)}$ is k -good, hence by Lemma 3.3.1 that $[\mathfrak{A}]_{(a_1, b_2)}$ is k -good as well. We therefore have $(a_1, b_1) \sim (a_2, b_2)$, which contradicts the fact that (a_1, b_1) and (a_2, b_2) belong to different equivalence classes of $\sim$. $\square$

Wrap Up

We have just established the following result.

Theorem 3.3.10. *For every (n, m) -chain $\mathfrak{A}$, with $n, m \geq 0$, and $k \geq 0$, there exists a constructible (n, m) -chain $\mathfrak{B}$ such that $\mathfrak{A} \equiv_k \mathfrak{B}$.*

This theorem admits the following important corollary.

Corollary 3.3.3. *A formula φ of $\text{MFO}(\mathbb{R}, <)$ with n free first-order and m free second-order variables, with $n, m \geq 0$, is satisfiable iff there exists a constructible (n, m) -chain $\mathfrak{A}$ such that $\mathfrak{A} \models \varphi$.*

Proof. One can apply Theorem 3.3.10 with k equal to the quantifier depth of the formula φ . $\square$

We are now ready to prove the main result of this paper.

Theorem 3.3.11. *Satisfiability is decidable for formulas in $\text{MFO}(\mathbb{R}, <)$.*

Proof. By Corollary 3.3.3, a semi-decision procedure can be obtained by enumerating all possible constructible chains, or more precisely the chain terms that we have introduced for describing sets of such chains, and checking for each of them whether the formula is satisfied.

To get full satisfiability, a criterion is needed for stopping this procedure at some point when the formula is not satisfiable. Let n and m be respectively the numbers of free first-order and second-order variables in the formula, and k be its quantifier depth. Using the results developed in Sections 3.3.3 and 3.3.3 together with Theorem 3.3.9, one can compute the set T of all possible k -types of constructible (n, m) -chains. This computation can be carried out by starting with the set of possible k -types of all constant and shuffle (n, m) -chains, and then recursively adding to this set the k -types of the chains that are obtained by applying reordering, concatenation, repetition or shuffle composition operators to chains contained in the set. Since the number of possible k -types of (n, m) -chains is finite by Theorem 3.3.1, this computation will eventually terminate. One can then check explicitly, for each $\tau \in T$, whether (n, m) -chains of k -type τ satisfy the formula, using the results of Section 3.3.3. $\square$

Chapter 4

Automata-Based Decision Procedures

In this chapter, we first recall the general principles of automata-based decision procedures for deciding the satisfiability of a given theory. We then specify how this approach can be adapted to deal with the monadic first-order theory of order over various classes of domains, using automata on linear orderings.

4.1 Illustration of the Approach

Overview The approach introduced by Büchi [22] to decide S1S consists in the following. Let T be the theory of interest. The first step is to define an encoding scheme for every possible interpretation in this theory. Then, every atom Λ of the theory T is turned into an automaton that accepts exactly the encodings of every model of Λ . The next step consists in defining operations on automata matching the logical operators of T . In that way, for any formula φ of T , one can inductively build an automaton $\mathcal{A}_\varphi$ that accepts exactly the language of the encodings of the models of φ , by following the syntactic structure of φ . Deciding the satisfiability of φ amounts to checking whether $\mathcal{A}_\varphi$ accepts a non-empty language.

The Case of S1S Let us illustrate this approach by briefly describing the decision procedure for S1S introduced by Büchi, relying on its eponymous automata. The theory S1S is interpreted over $\mathbb{N}$ and involves both first-order and second-order variables. An interpretation of a formula $\varphi(x_1, \dots, x_n, P_1, \dots, P_m)$ can be encoded as an ω -word over the alphabet $\Sigma = \{0, 1\}^{n+m}$. Such a word can be viewed as the content of $n + m$ tapes each supporting an ω -word over the alphabet $\{0, 1\}$. For all $i \in [1, n]$, the i -th tape encodes the value of the first-order variable x_i in *unary notation*. In other words, if the variable x_i takes the value $\ell \in \mathbb{N}$, then the corresponding i -th tape contains the word $0^\ell \cdot 1 \cdot 0^{+\omega}$, i.e., the word $(\alpha_k)_{k \in \mathbb{N}}$ defined as $\alpha_\ell = 1$, and $\alpha_k = 0$ for all $k \in \mathbb{N} \setminus \{\ell\}$. For all

$j \in [n+1, n+m]$, the j -th tape encodes the value of the second-order variable P_{n-j} also in unary notation, i.e., the subset $N \subseteq \mathbb{N}$ associated with P_{n-j} is represented by the ω -word $(\alpha_k)_{k \in \mathbb{N}}$ defined by $\alpha_k = 1$ if $k \in N$, and $\alpha_k = 0$ if $k \notin N$, for all $k \in \mathbb{N}$.

Atomic formulas in S1S are of the form $x = y$, $x = y + 1$, or $P(x)$, where x and y are first-order variables, and P is a second-order variable. Büchi automata corresponding to these atoms are described in Figure 4.1. We use the symbol $*$ to denote any symbol from the alphabet $\{0, 1\}$. Since the encoding of a first-order variable can only contain a single occurrence of the symbol 1, in order to make the notation lighter, we may discard the tape associated to a first-order variable x entirely from the depicted automaton, and simply place the symbol x on the transitions supposed to be labeled with a 1. An alternate representation of the automaton for the atom $P(x)$, with this variant notation, is depicted in Figure 4.2.

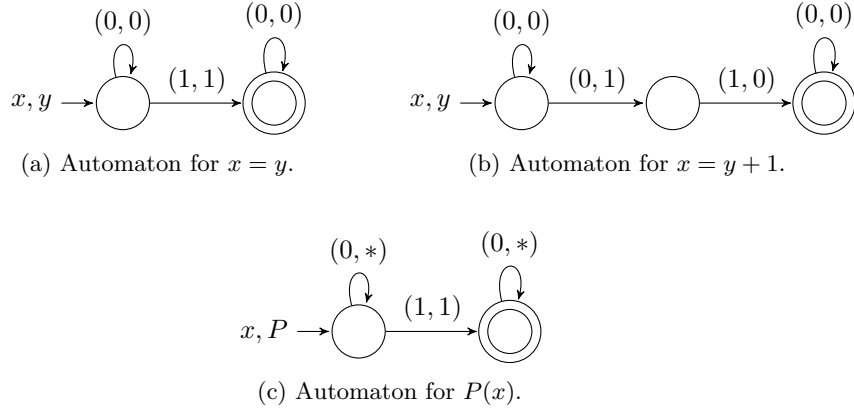


Figure 4.1: Automata for some atoms of S1S.

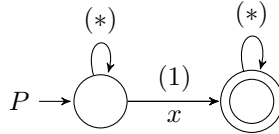


Figure 4.2: Alternate version of the automaton for $P(x)$.

Let us now describe operations on Büchi automata matching the logical operators $\vee$, $\exists$, and $\neg$. Since the operators $\wedge$ and $\forall$ can be expressed in terms of others, dealing with them explicitly is unnecessary. Indeed, formulas of the form $\varphi_1 \wedge \varphi_2$ and $\forall x. \varphi$ can be respectively rewritten as $\neg(\neg\varphi_1 \vee \neg\varphi_2)$ and $\neg(\exists x. \neg\varphi)$.

The disjunction $\vee$ can be handled using automata union. More precisely, an automaton for the formula $\varphi_1 \vee \varphi_2$ can be obtained from automata $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$, respectively recognizing the models of φ_1 and φ_2 , in two steps:

- First, unify the alphabets of the two automata $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ so that their shared updated alphabet accounts for all the free variables of both φ_1 and φ_2 , in the same order. Let $\{x_1, \dots, x_n, P_1, \dots, P_m\}$ be the set of free variables occurring in either φ_1 , φ_2 , or both. The unification process yields two automata $\mathcal{A}'_{\varphi_1}$ and $\mathcal{A}'_{\varphi_2}$ defined over the common alphabet $\Sigma' = \{0, 1\}^{n+m}$ such that, intuitively, for any word u labelling an accepting path in either $\mathcal{A}'_{\varphi_1}$ or $\mathcal{A}'_{\varphi_2}$, the projection of u over its k -th tape encodes the value of the free first-order variable x_k if $k \in [1, n]$, or the second-order variable P_{k-n} if $k \in [n+1, n+m]$.
- Second, compute the union automaton $\mathcal{A}'_{\varphi_1} \cup \mathcal{A}'_{\varphi_2}$. (The union of Büchi automata can be computed analogously to the union of automata over linear orderings introduced in Section 1.3.2.)

Existential quantification is also easy to deal with, and the corresponding operation is simply a projection: the tape associated to the quantified (first or second-order) variable is erased. The operation on automata corresponding to the negation of a formula is close to a set difference operation, i.e., given an input formula φ and an automaton $\mathcal{A}_\varphi$ recognizing its models, one builds an automaton accepting all the words rejected by $\mathcal{A}_\varphi$ that still correspond to valid encodings of the free variables of φ . This operation can be performed by first complementing the automaton $\mathcal{A}_\varphi$, and then intersect the resulting automaton with elementary automata enforcing that each tape associated with a first-order variable follows exactly once a transition labeled with the symbol 1.

Finally, the last step consists in checking if the language accepted by the resulting Büchi automaton is non-empty, which is a decidable operation [22, 115].

In the decision procedure for S1S we just described, the only computationally costly operation is complementation. Complementing non-deterministic Büchi automata is not a trivial operation, and a lot of effort have been put into designing efficient complementation algorithms [102, 71, 118].

Actual implementations of automata-based decision procedures elude this problem by restricting the language of interest or the class of automata that need to be manipulated. For instance, the tool MONA [69] only handles *Weak* S1S (WS1S) which is, schematically, a restriction of S1S to finite subsets of natural number [21]. The tool LASH [16] handles the mixed integer and real additive arithmetic by working with weak deterministic automata, which are a restricted form of infinite-word automata admitting an easy complementation algorithm [15].

Variant for LTL It is worth mentioning that this approach can also be adapted to decide some temporal logics [108]. In this context, one can define suitable operations on automata to match the different modalities involved in the temporal logic being considered. For instance, Cristau [38] provided a translation of the classical and Stavi modalities (required to deal with incom-

plete orderings) into automata on linear orderings¹.

4.2 Deciding MFO($<$)

In order to deal with the monadic theory of order over various classes of orderings (the single domains $\mathbb{R}$ and $\mathbb{Q}$, the class $\mathcal{O}$ of all ordinals, the class $\mathcal{L}$ of all linear orderings, ...) it is clear that Büchi automata are not adapted. Our approach, which we precisely describe thereafter, is heavily inspired by the procedure sketched for S1S, but relies on automata on linear orderings instead of Büchi automata.

We make the choice to keep most of the procedure as general as possible, in particular by being oblivious to the class of domains over which we intend to interpret the theory of order for as long as possible. More precisely, our approach consists in, given an input formula φ , building an automaton $\mathcal{A}_\varphi$ that recognizes the models of the formula φ over the class $\mathcal{L}$ of all linear orderings. We delay the filtering by domains to the last step, by designing non-emptiness tests dedicated to the classes of orderings we are interested in. The details and motivations of this choice in our approach will be presented in Chapter 5.

Encoding Variables Given a linear ordering J , one can encode any element $j \in J$ and any subordering $J' \subseteq J$ by words indexed by linear orderings. More precisely, an element $j \in J$ is encoded using *unary notation* by the word $0^{J^{<j}} \cdot 1 \cdot 0^{J^{>j}}$ indexed by J and defined over the alphabet $\{0, 1\}$, where $J^{<j} = \{k \in J \mid k < j\}$ and $J^{>j} = \{k \in J \mid j < k\}$. Similarly, a subordering $J' \in J$ is encoded by the word $(\alpha_j)_{j \in J}$ such that $\alpha_j = 1$ if $j \in J'$, and $\alpha_j = 0$ otherwise.

Let J be a linear ordering, $j_1, \dots, j_n \in J$ be $n \geq 0$ elements, and $J_1, \dots, J_m \subseteq J$ be $m \geq 0$ suborderings. The tuple $(j_1, \dots, j_n, J_1, \dots, J_m)$ is encoded by combining the individual encoding of each element and subordering into a single word over the alphabet $\Sigma = \{0, 1\}^{n+m}$. This word can be seen as the contents of $n+m$ parallel tapes each defined over the alphabet $\{0, 1\}$, such that the i -th tape corresponds to the element j_i if $i \in [1, n]$, or to the subordering J_{i-n} if $i \in [n+1, n+m]$.

Consider the mappings defined as follows:

$$\Theta_{\neq i} : \Sigma^\ell \rightarrow \Sigma^{\ell-1} : (\alpha_1, \dots, \alpha_\ell) \mapsto (\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_\ell)$$

In the following, we use the notation $\Theta_{\neq i}(u)$, where u is a word indexed by a linear ordering, to express the result of applying $\Theta_{\neq i}$ to each symbol in u , that is, if $u = (\beta_j)_{j \in J}$, then $\Theta_{\neq i}(u) = (\Theta_{\neq i}(\beta_j))_{j \in J}$. For a language L , we write $\Theta_{\neq i}(L)$ in place of the language $\{\Theta_{\neq i}(u) \mid u \in L\}$. Analogously, we also define the projections $\Theta_{=i} : \Sigma^\ell \rightarrow \Sigma : (\alpha_1, \dots, \alpha_\ell) \mapsto \alpha_i$.

¹The automata considered are slightly different from the ones we consider in the sense that they are transducers, i.e., they have an output, and are not simply accepting or rejecting.

(n, m)-Automata An (n, m) -automaton on linear orderings $\mathcal{A}$ for $n, m \geq 0$, is an automaton defined over the alphabet $\{0, 1\}^{n+m}$, such that for every word $u \in L(\mathcal{A})$ and every $i \in [1, n]$, the word $\Theta_{=i}(u)$ is of the form $0^J \cdot 1 \cdot 0^K$ for some orderings J and K , and the orderings $J + 1 + K$ and $|u|$ are order-isomorphic. In other words, every accepting path of $\mathcal{A}$ follows exactly one successor transition labeled by a symbol of the form $(\alpha_0, \dots, \alpha_{i-1}, 1, \alpha_{i+1}, \dots, \alpha_{n+m})$ for some $\alpha_1, \dots, \alpha_{n+m} \in \{0, 1\}$, for every $i \in [1, n]$. We say that a language L is a (n, m) -language if there exists an (n, m) -automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.

Intuitively, in an (n, m) -automaton, the n first tapes encode n first-order variables $x_1, \dots, x_n$, while the m last tapes encode m second-order variables $P_1, \dots, P_m$. For these reasons, in an (n, m) -automaton we will refer to the i -th tape as either x_i if $i \in [1, n]$, or as P_{i-n} if $i \in [n+1, n+m]$.

Consider an (n, m) -automaton $\mathcal{A}$, and two *useful* states q and q' of $\mathcal{A}$, i.e., states that are visited by at least one accepting run of $\mathcal{A}$. Then the following property holds.

Property 4.2.1. *If $\mathcal{A}$ admits a successor transition associated with a first-order variable x_i from the state q to the state q' , then $\mathcal{A}$ cannot admit another transition from q to q' that is not associated with x_i .*

Proof. By contradiction, assume that $\mathcal{A}$ admits two distinct successor transitions from q to q' , such that one is associated with x_i , but the other is not. Then, starting from two accepting runs visiting respectively q and q' , one could easily build a new accepting run that either never follows a transition associated with x_i , or follows two such transitions. $\square$

Complementation Before turning to the translation of the atoms of MFO(<) into automata, let us address the elephant in the room: in the general case, automata on linear orderings are not closed under complementation [5]. In other words, the complement of a regular language (defined over the class of all linear orderings) may not be regular. Two identified limits for complementation are presented below.

Lemma 4.2.1 ([5]). *For any given finite alphabet Σ , let L_Σ^S denote the language of all words indexed by scattered linear orderings over Σ . We denote by $(L_\Sigma^S)^C$ its complement, i.e., the language of all words indexed by non-scattered linear orderings over Σ . The language $(L_\Sigma^S)^C$ is regular, while L_Σ^S is not.*

Proof sketch. An automaton recognizing the language $(L_\Sigma^S)^C$ is depicted in Figure 4.3. The principle is that in order to reach the accepting state q_2 from the initial state q_1 , the only possibility is to follow limit transitions involving the set of states $\{q_1, q_2\}$. Such transitions can only be followed within a shuffle structure, where occurrences of cuts mapped to q_1 and q_2 are densely distributed.

Let us give the intuition behind the fact that the language L_Σ^S is not regular. If there existed an automaton $\mathcal{A}_S$ accepting exactly the language L_Σ^S , then it would notably accept words indexed by orderings of the form ζ^n (where ζ is the order type of the ordering $\mathbb{Z}$), for any $n \in \mathbb{N}$. For a sufficiently large value

of n (with respect to the number of states in $\mathcal{A}_S$), it can be shown that the automaton $\mathcal{A}_S$ necessarily possesses a left-limit transition of the form $P \rightarrow q$, and a right-limit transition of the form $p \rightarrow P$, with $p, q \in P$, such that each of these transitions is followed in at least one accepting paths in $\mathcal{A}_S$. One can then build an accepting path in $\mathcal{A}_S$ labeled by a non-scattered word, where the underlying shuffle structure is read thanks to the transitions $P \rightarrow q$ and $p \rightarrow P$, which yields a contradiction. $\square$

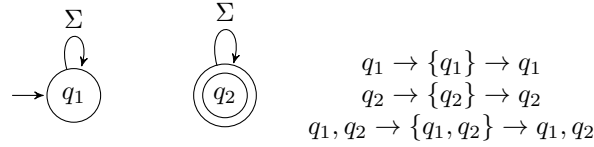


Figure 4.3: An automaton accepting every word indexed by a non-scattered ordering.

Lemma 4.2.2. *For any given finite alphabet Σ , let $L_\Sigma^{\text{Cof} \leq \omega}$ denote the language of all words indexed by ordinals of cofinality less than or equal to ω over Σ , and let $L_\Sigma^{\text{Cof} > \omega}$ denote the language of all words indexed by ordinals of cofinality strictly greater than ω . The language $L_\Sigma^{\text{Cof} \leq \omega}$ is regular, while $L_\Sigma^{\text{Cof} > \omega}$ is not.*

Notice that $L_\Sigma^{\text{Cof} > \omega}$ is the complement of $L_\Sigma^{\text{Cof} \leq \omega}$ only over the class of words indexed by ordinals, but not over the class of all linear orderings. Enforcing that an ordering contains at least one successor-less cut in order to cover all orderings in $\mathcal{L} \setminus \mathcal{O}$ is easy to express using automata, by requiring that at least one right-limit transition is followed. It follows that the complement of the language $L_\Sigma^{\text{Cof} \leq \omega}$ over the class of all linear orderings cannot be regular, otherwise $L_\Sigma^{\text{Cof} > \omega}$ would also be regular.

Proof sketch. An automaton (adapted from Example 18 in [4]) recognizing the language $L_\Sigma^{\text{Cof} \leq \omega}$ is depicted in Figure 4.4. A first observation is that this automaton can only accept words indexed by ordinals since it only contains left-limit transitions. Any accepting path in the automaton starts in either q_1 or q_3 , and ends in either q_3 or q_4 . Any path that ends in q_3 (regardless of whether it starts in q_1 or q_3) necessarily admits a finite suffix visiting only q_3 , and is therefore indexed by a successor ordinal, hence of cofinality 1. Paths that ends in q_4 by following the limit transition $\{q_3\} \rightarrow q_4$ are necessarily indexed by an ordinal of the form $\nu \cdot \omega$ (where ν is any ordinal, even an uncountable one), since the automaton does not contain a limit transition $\{q_3\} \rightarrow q_3$ to allow for greater limit ordinals while only visiting the state q_3 . Finally, the only remaining case to consider is an accepting path that ends in q_4 after following the limit transition $\{q_1, q_2\} \rightarrow q_4$. By contradiction, let us assume that there exists an accepting path π labeled by a word u indexed by an ordinal of cofinality strictly greater than ω . Since we have already ruled out the other possibilities, we

furthermore assume that it ends by following the limit transition $\{q_1, q_2\} \rightarrow q_4$. Let $C_{q_1} = \{c \in \widehat{u} \mid \pi(c) = q_1\}$ be the set of cuts in $\widehat{u}$ at which the path π visits the state q_1 . The set C_{q_1} is necessarily cofinal in $\widehat{u}$ by definition of π , and is therefore of cardinality strictly greater than $\aleph_0$. It follows that there exists a sequence $(c_i)_{i \in \mathbb{N}}$ of cuts in C_{q_1} that converges towards a cut $d \in \widehat{u}$ which is not the last cut $c_{\max}$. The cut d is necessarily a predecessor-less cut, and must be reached by following a limit transition. The definition of the sequence of cuts $(c_i)_{i \in \mathbb{N}}$ also entails that $\lim_d \pi = \{q_1, q_2\}$. But since the only left-limit transition involving the limit set $\{q_1, q_2\}$ is the transition $\{q_1, q_2\} \rightarrow q_4$, it would imply that $\pi(d) = q_4$. There is however no way to exit state q_4 , which means that the path π over the interval $[d, c_{\max}]$ cannot be valid, since $d \neq c_{\max}$. Therefore, we have a contradiction, and such an accepting path π simply cannot exist.

The intuition behind the fact that $L_{\Sigma}^{\text{Cof} > \omega}$ is not regular is that any automaton accepting words indexed by uncountable ordinals necessarily also accepts words indexed by countable ordinals. More precisely, to accept a word indexed by an uncountable ordinal, an automaton on linear orderings relies on left-limit transitions of the form $P \rightarrow q$ with $q \in P$. But it is always possible to build paths indexed by countable ordinals involving the same left-limit transitions. (The description of reachability algorithms for automata on linear orderings, as presented in Chapter 5, provides more insight on this matter.)

As a side note, let us remark that this property does not hold over all linear orderings. For instance, the automaton depicted in Figure 1.5 and presented in Example 7 — which accepts the language associated with the regular expression $\text{sh}(a, b)$ — only accepts words indexed by infinite, dense, and complete orderings. These three constraints combined can only be satisfied by an uncountable ordering. $\square$

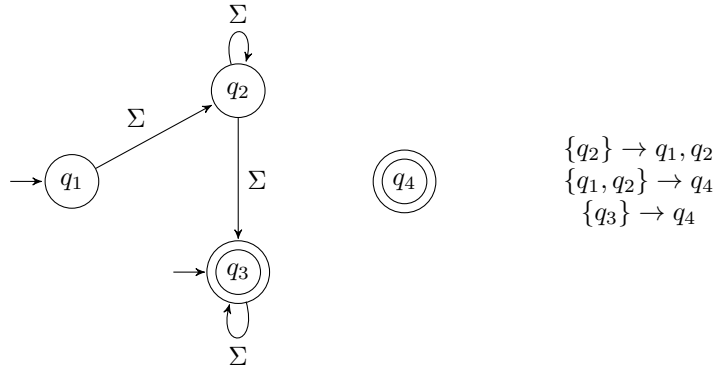


Figure 4.4: An automaton accepting every word indexed by an ordinal of cofinality less than or equal to ω .

It is worth mentioning that, over the class of countable and scattered linear orderings, regular languages are closed under complementation [98, 97], and

an effective complementation algorithm can be derived from the proofs in [98] and [36]. The resulting complementation construction is based on an algebraic approach, and uses semigroup theory. This algebraic approach has been further developed to deal with related problems in [30] and [17]. In this specific context, the decision procedure sketched for S1S can be straightforwardly adapted to decide the monadic second-order theory of order over countable and scattered linear orderings, as presented in [5].

However, over the class of all linear orderings, we have to find another way to deal with negation, conjunction, and universal quantification. Recall that we are interested in deciding the monadic first-order theory of order, and that only first-order universal quantification must therefore be handled. Any formula in $\text{MFO}(<)$ can easily be put in a prenex normal form with the additional constraint that every negation is pushed inward so as to only affect atomic subformulas. This alternative requires to deal with conjunction and first-order universal quantification head-on, but as long as the negation of each atomic formula admits a corresponding automaton, no complementation needs to be performed.

Atoms (and their Negations) to Automata We are now ready to define an automaton for each atom, and its negation, of $\text{MFO}(<)$, over the class of all linear orderings. It is sufficient to consider the following set of atoms: $\{P(x), x < y\}$. The automata associated with the atoms and their negations are depicted in Figure 4.5.

Operations on Automata Let us now turn to the definition of the proper operations on automata on linear orderings to match the Boolean connectives $\wedge$ and $\vee$, and the first-order quantifiers $\exists$ and $\forall$. For disjunction and conjunction, we rely on automata product. Namely, disjunction is dealt with automata union, and conjunction with automata intersection. These operations have been defined in Section 1.3.2.

Similarly as in the decision procedure for S1S, before computing the union or intersection of automata, one however needs to unify their alphabets. We only explicitly deal with the disjunctive case, the conjunctive case can be managed identically. Consider a disjunctive formula of the form

$$\begin{array}{c} \varphi_1(x_1, \dots, x_n, y_1, \dots, y_{n'}, X_1, \dots, X_m, P_1, \dots, P_{m'}) \\ \vee \\ \varphi_2(x_1, \dots, x_n, z_1, \dots, z_{n''}, X_1, \dots, X_m, R_1, \dots, R_{m'') \end{array}$$

and let $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ be automata respectively recognizing the models of φ_1 and φ_2 , obtained via the construction we are describing. By construction, the automaton $\mathcal{A}_{\varphi_1}$ is defined over the alphabet $\{0, 1\}^{n+n'+m+m'}$, where the $n+n'$ first tapes are associated with first-order variables, and the $m+m'$ last tapes are associated with second-order variables. Similarly, the automaton $\mathcal{A}_{\varphi_2}$ is defined over the alphabet $\{0, 1\}^{n+n''+m+m''}$. The automaton $\mathcal{A}_{\varphi_1 \vee \varphi_2}$ recognizing the models of $\varphi_1 \vee \varphi_2$ must be defined over the alphabet $\{0, 1\}^{n+n'+n''+m+m'+m''}$

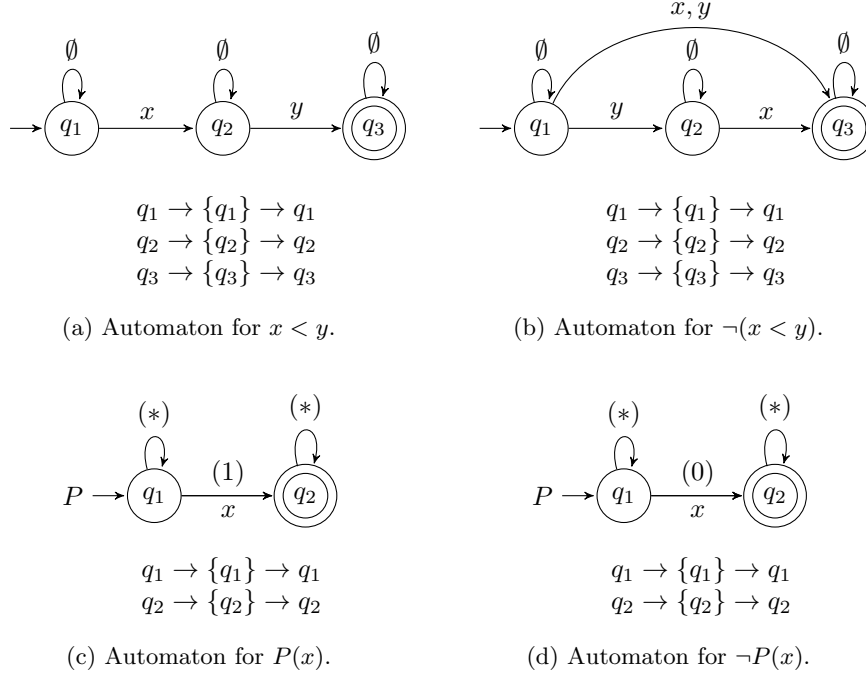


Figure 4.5: Automata for the atoms of MFO(<) and their negations.

in order to encode every free variable of both φ_1 and φ_2 . Our approach consists in extending the alphabet of $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ up to $\{0, 1\}^{n+n'+n''+m+m'+m''}$. Let us detail the case of $\mathcal{A}_{\varphi_1}$. We need to incorporate n'' new tapes dedicated to first-order variables, and m'' new tapes dedicated to second-order variables.

Let us describe, in a general manner, how to incorporate an additional tape dedicated to encode a fresh second-order variable R in a (j, k) -automaton $\mathcal{A} = (Q, \{0, 1\}^{j+k}, I, F, \Delta^s, \Delta^\ell)$. Since there is no restriction on the value of the interpretation of R , this can be done by extending the alphabet, and turning each successor transition in Δ^s into two new ones, one for each possible truth value of the interpretation of R at every given position. Formally, we build the automaton $\mathcal{A}_{+SO} = (Q, \{0, 1\}^{j+k+1}, I, F, \Delta_{+SO}^s, \Delta^\ell)$ similar to $\mathcal{A}$ except for its extended alphabet, and its set of successor transitions Δ_{+SO}^s defined as

$$\Delta_{+SO}^s = \{(q, (\alpha_1, \dots, \alpha_{j+k}, b), p) \mid (q, (\alpha_1, \dots, \alpha_{j+k}), p) \in \Delta^s \wedge b \in \{0, 1\}\}.$$

Notice that iteratively incorporating ℓ fresh second-order tapes to a (j, k) -automaton $\mathcal{A}$ implies that each original successor transitions of $\mathcal{A}$ is turned into 2^ℓ new successor transitions.

Incorporating a new tape dedicated to a fresh first-order variable z is more involved. One has to enforce that this new tape encodes a first-order variables, i.e., that exactly one occurrence of the symbol 1 is read on this tape in each accepting path. It can be managed by first considering the fresh tape as an un-

restricted second-order tape, and then intersect the augmented automaton with another very simplistic automaton enforcing the first-order constraint. Such an automaton, defined over the alphabet $\{0, 1\}^{k+1}$, enforcing that its $(k+1)$ -th tape encodes a first-order variable is denoted by $\mathcal{C}_{k+1}^{\text{FO}}$, and is depicted in Figure 4.6. The label $(\ast^k, b)$ represents any $(k+1)$ -tuple in $\{0, 1\}^{k+1}$ where the $(k+1)$ -th component is set to the fixed value $b \in \{0, 1\}$. Starting from a (j, k) -automaton $\mathcal{A}$, the addition of a fresh tape associated with an unrestricted first-order variable to $\mathcal{A}$ amounts to building the automaton $\mathcal{A}_{+\text{FO}}$ defined as $\mathcal{A}_{+\text{FO}} = \mathcal{A}_{+\text{SO}} \cap \mathcal{C}_{j+k+1}^{\text{FO}}$.

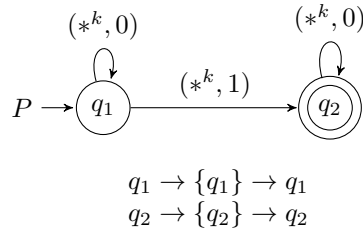


Figure 4.6: The automaton $\mathcal{C}_{k+1}^{\text{FO}}$ enforcing the $(k+1)$ -th tape to encode a first-order variable.

To complete the unification of the alphabets, some reordering of the tapes may be required, so as to ensure that for any index $k \in [1, n + n' + n'' + m + m' + m'']$, the k -th tape supports the encoding of the same variable in both automata. Reordering the tapes is an easy task. It amounts to, for a given permutation (i.e., a bijection from a set to itself) σ on the set $\{1, 2, \dots, n + n' + n'' + m + m' + m''\}$, changing the label of every successor transition as follows: any label of the form $(\alpha_1, \alpha_2, \dots, \alpha_{n+n'+n''+m+m'+m''})$ is replaced by the label $(\alpha_{\sigma(1)}, \alpha_{\sigma(2)}, \dots, \alpha_{\sigma(n+n'+n''+m+m'+m'')})$.

Let $\mathcal{A}'_{\varphi_1}$ and $\mathcal{A}'_{\varphi_2}$ denote the two automata obtained from $\mathcal{A}_{\varphi_1}$ and $\mathcal{A}_{\varphi_2}$ after all the required additional tapes have been incorporated and reordered accordingly. An automaton $\mathcal{A}_{\varphi_1 \vee \varphi_2}$ recognizing the models of the disjunctive formula $\varphi_1 \vee \varphi_2$ can then be obtained by computing the union automaton $\mathcal{A}'_{\varphi_1} \cup \mathcal{A}'_{\varphi_2}$ following the procedure described in Section 1.3.2. The same principle can be directly applied to deal with formulas of the form $\varphi_1 \wedge \varphi_2$.

First-order existential quantification simply amounts to a projection, i.e., erasing the tape associated to the first-order variable being quantified, exactly as in the decision procedure for S1S. First-order universal quantification on the other hand, is much more involved, and is the crux of the entire decision procedure. First-order universal quantification of automata will be addressed in Chapter 6.

Non-Emptiness Once the entire input formula has been processed, and a corresponding automaton has been obtained, the last step of the procedure

consists in checking whether the language accepted by the automaton is non-empty.

Since at this point the decision procedure is completely oblivious of the domain (or class of domains) over which we interpret formulas, it will be the role of the non-emptiness test to properly filter the models that can fit the intended domains. The design of domain-specific non-emptiness tests is addressed in Chapter 5.

Chapter 5

Non-Emptiness Test

The *non-emptiness* decision problem takes an automaton as input, and consists in deciding whether the language accepted by this automaton contains at least one word, i.e., whether $L(\mathcal{A}) \neq \emptyset$ holds. The non-emptiness problem is strongly related to the *reachability* (or *accessibility*) decision problem, which, given an automaton $\mathcal{A}$ and two of its states q and p , consists in determining whether $\mathcal{A}$ admits a path starting in q and ending in p . Reachability for a triple $(\mathcal{A}, q, p)$ reduces to non-emptiness on an automaton $\mathcal{A}'$, which is identical to $\mathcal{A}$ except for its sets of initial and accepting states: the only initial state of $\mathcal{A}'$ is q , and its only accepting state is p . Non-emptiness for an automaton $\mathcal{A}$ reduces to reachability on all triples of the form $(\mathcal{A}, q_I, q_F)$, where q_I is an initial state of $\mathcal{A}$, and q_F is an accepting state of $\mathcal{A}$. For these reasons, we may use the notions of reachability (or accessibility) and non-emptiness interchangeably.

For automata on finite words, the non-emptiness problem directly reduces to the accessibility problem in a directed graph. The only transitions are successor transitions, and their labels do not play any role in determining the existence of a path. For Büchi automata, deciding non-emptiness can be performed by a combination of checking accessibility in the underlying directed graph, combined with the detection of cycles involving at least one accepting state. The same principle applies for Muller automata, except that the constraint on the cycles is not that it contains an accepting state but that the set of states visited by the cycle coincides exactly with an accepting component. (Recall that the accepting condition of a Muller automaton does not involve individual accepting states, but accepting components which correspond to sets of states, cf. Section 1.3.2.)

Let us now investigate how non-emptiness can be decided for automata on linear orderings, and more specifically when the domain is set as either $\mathbb{R}$ or $\mathbb{Q}$, i.e., how to decide whether a given input automaton on linear orderings admits an accepting path labeled by a word indexed by either $\mathbb{R}$ or $\mathbb{Q}$.

The remainder of this chapter reproduces the core — i.e., Sections 3 to 6 — of the following article [12]:

Bernard Boigelot, Pascal Fontaine, **Baptiste Vergain**. Deciding reachability in automata on words indexed by the reals and rationals. *Theoretical Computer Science*, 1080:116048, 2026.

The introduction, preliminaries, and conclusion sections have been excluded to avoid redundancy. We additionally give a few remarks not initially present in the article at the end of this chapter.

5.1 State of the Art

Given an automaton on linear orderings $\mathcal{A}$ and two of its states p and q , the reachability problem consists in deciding whether $\mathcal{A}$ admits a path starting at p and ending in q . In [29], Carton describes a polynomial procedure for deciding reachability on scattered linear orderings only, i.e., whether there exists a path labeled by a word indexed by such an ordering. Given the input automaton $\mathcal{A}$, Carton’s algorithm generates a finite-word automaton $\mathcal{A}_F$ sharing the same set of states as $\mathcal{A}$, that accounts for every possible path in $\mathcal{A}$ while storing in the labels read along the paths the set of states that are visited. More precisely, the existence of a transition (p, P, q) in $\mathcal{A}_F$ implies that there exists in $\mathcal{A}$ a path $p \rightarrow q$ labeled by a word indexed by a scattered linear ordering that visits exactly the set of states P . Reciprocally, the existence in $\mathcal{A}$ of a path $p \rightarrow q$ labeled by a word indexed by a scattered linear ordering implies the existence of a matching path $p \rightarrow q$ in $\mathcal{A}_F$, although it may not necessarily be composed of a single transition. By not explicitly building a transition for every possible path, Carton’s solution remains polynomial in the size of the automaton. Reachability in $\mathcal{A}$ then reduces to reachability in $\mathcal{A}_F$. This algorithm can be used to solve also the non-emptiness problem over scattered linear orderings, by checking whether an accepting state is reachable from an initial state.

A generalization of this algorithm over arbitrary linear orderings is sketched in [38], where the author introduces a specific additional rule for dealing with shuffles. To the best of our knowledge, a proof of this generalization has not been published, and it seems that the additional rule is actually only able to deal with complete shuffles. Actually, a small modification of this rule (somewhat similar to what will be done in Section 5.3) is enough to deal with both complete and incomplete shuffles altogether, so as to accurately cover all linear orderings.

5.2 Reachability over the Reals

5.2.1 Characterization

Many properties characterizing $\mathbb{R}$, such as completeness, density, and not having a first or last element, can easily be expressed by automata on linear or-

derings. One could therefore hope that a procedure to check the non-emptiness (or equivalently decide the reachability problem) of an automaton over $\mathbb{R}$ may be designed by intersecting [38] this automaton with automata describing sufficiently many of the properties of $\mathbb{R}$, and then check the non-emptiness of the resulting automata.

Unfortunately, as shown by the following result, if accepting words indexed by complete and dense orderings without first or last element is a necessary condition for accepting words indexed by $\mathbb{R}$, this criterion is not sufficient.

Theorem 5.2.1. *There exists an automaton on linear orderings $\mathcal{A}_{>\mathbb{R}}$ that accepts words indexed by non-empty, dense, and complete linear orderings that do not have a first or last element, and that does not accept any word indexed by $\mathbb{R}$.*

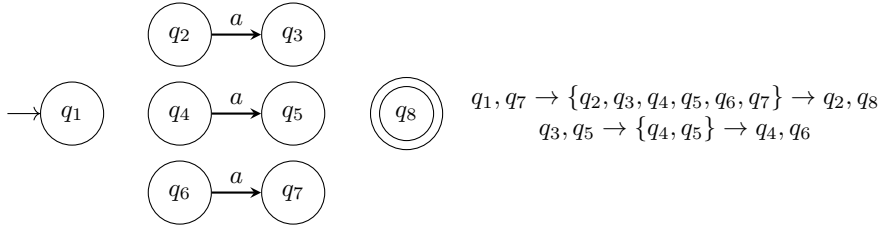


Figure 5.1: Automaton $\mathcal{A}_{>\mathbb{R}}$ that does not accept any word indexed by $\mathbb{R}$.

The automaton $\mathcal{A}_{>\mathbb{R}}$ depicted in Figure 5.1 matches the description given in Theorem 5.2.1. The proof of Theorem 5.2.1 ensues from the two following lemmas.

Lemma 5.2.1. *The automaton $\mathcal{A}_{>\mathbb{R}}$ does not accept any word indexed by $\mathbb{R}$.*

Proof. Let w be a word accepted by $\mathcal{A}_{>\mathbb{R}}$. We first prove that the ordering $|w|$ must be of the form $\Sigma_{k \in K} J_k$, where for all $k \in K$, J_k is an infinite, dense, and complete ordering with first and last elements, and K is an infinite, dense, and complete ordering without a first or last element. Intuitively, each J_k corresponds to a subpath that starts in q_2 , continues in q_3 , then visits a dense combination of q_4 and q_5 before reaching q_6 , and finally ends in q_7 .

The absence of first and last elements in K (and hence in $|w|$) results from the fact that the unique initial state q_1 does not admit an outgoing successor transition, and the unique accepting state q_8 does not admit an incoming successor transition. On the other hand, each J_k admits a first element that corresponds to the successor transition (q_2, a, q_3) , and a last element that corresponds to the successor transition (q_6, a, q_7) .

Furthermore, since there does not exist a state q in $\mathcal{A}_{>\mathbb{R}}$ that admits both limit transitions $P \rightarrow q$ and $q \rightarrow P'$, it is impossible to map a gap of an incomplete ordering to a state of $\mathcal{A}_{>\mathbb{R}}$, hence $|w|$, and therefore K , as well as every J_k , are necessarily complete.

Recall that a linear ordering is dense if and only if it does not admit two consecutive elements. The density of K and each J_k naturally ensues from the fact that there does not exist a state q in $\mathcal{A}_{>\mathbb{R}}$ that admits both an incoming successor transition $p \rightarrow q$ and an outgoing successor transition $q \rightarrow p'$.

We now show that K and every J_k are infinite. By definition, if a path π in some automaton on linear orderings follows a limit transition, then an infinite set of states is necessarily visited in π . It ensues that π follows an infinite set of successor transitions, and that the word labeling π must be indexed by an infinite linear ordering. In our case, every subpath from q_2 to q_7 necessarily follows the limit transitions $q_3, q_5 \rightarrow \{q_4, q_5\} \rightarrow q_4, q_6$. Therefore, every J_k is infinite. Moreover, the only possibility of leaving (resp. reaching) the state q_1 (resp. q_8) is to follow the limit transition $q_1 \rightarrow \{q_2, q_3, q_4, q_5, q_6, q_7\}$ (resp. $\{q_2, q_3, q_4, q_5, q_6, q_7\} \rightarrow q_8$). It follows that K is also infinite.

The ordering $\mathbb{R}$ cannot however be expressed as an ordering $\Sigma_{k \in K} J_k$ such that K is an infinite, dense, and complete linear ordering without a first or a last element, and such that each J_k is an infinite, dense, and complete linear ordering with first and last elements. By contradiction, assume that $\mathbb{R}$ can be partitioned into an infinite, dense, and complete set I of pairwise disjoint closed intervals $[a, b]$ with $a < b$. The density of I implies that between any two intervals $[a, b], [e, f] \in I$ with $b < e$, there exists $[c, d] \in I$ such that $b < c < d < e$.

We first show that I must be uncountable. Otherwise, it would mean that I is both countable and complete. This contradicts Cantor's isomorphism theorem [28] which states that any infinite, dense, and countable ordering without first and last elements is order-isomorphic to $\mathbb{Q}$, and therefore incomplete.

Then, we show that I must be countable. Any set of disjoint intervals of $\mathbb{R}$ with non-zero length (such as I) is necessarily countable [106], since each of them contains a rational value that does not belong to the others.

Since I cannot be both countable and uncountable, we have a contradiction. $\square$

Lemma 5.2.2. *The automaton $\mathcal{A}_{>\mathbb{R}}$ accepts a word indexed by a non-empty, dense, and complete linear ordering that does not have a first or a last element.*

Proof. Consider the linear ordering $S = [0, 1] \cdot \mathbb{R}$, where $[0, 1]$ denotes the closed interval of $\mathbb{R}$ between 0 and 1. Intuitively, S corresponds to the real line in which every real number has been replaced by a copy of the closed interval $[0, 1]$. The ordering S is dense, complete, and does not admit a first or a last element. We can represent elements of S as pairs $(x, y) \in [0, 1] \times \mathbb{R}$, ordered by the relation $<_S$ defined by $(x_1, y_1) <_S (x_2, y_2)$ if either $y_1 < y_2$, or $y_1 = y_2$ and $x_1 < x_2$, where $<$ denotes the usual order relation on both $\mathbb{R}$ and $[0, 1]$. Now, consider the word w indexed by S (i.e., $|w| = S$) obtained by labeling each element of S with the symbol a . We show that there exists an accepting run of $\mathcal{A}_{>\mathbb{R}}$ reading w . We denote by $\hat{s}_{min}$ (resp. $\hat{s}_{max}$) the first (resp. last) cut of S . Consider the mapping $\pi : \hat{S} \rightarrow Q$ defined as follows:

- $\pi(\hat{s}_{min}) = q_1$ and $\pi(\hat{s}_{max}) = q_8$,

- for all $y \in \mathbb{R}$, $\pi((0, y)^-) = q_2$ and $\pi((0, y)^+) = q_3$,
- for all $y \in \mathbb{R}$, $\pi((1, y)^-) = q_6$ and $\pi((1, y)^+) = q_7$,
- for all $x \in (0, 1)$ and for all $y \in \mathbb{R}$, $\pi((x, y)^-) = q_4$,
- for all $x \in (0, 1)$ and for all $y \in \mathbb{R}$, $\pi((x, y)^+) = q_5$.

It is immediate to establish that π describes an accepting run. $\square$

This concludes the proof of Theorem 5.2.1.

5.2.2 Algorithm

We now introduce an algorithm that decides reachability over $\mathbb{R}$. Let $\mathcal{A} = (Q, \Sigma, I, F, \Delta^s, \Delta^\ell)$ be an automaton on linear orderings, $q_I, q_F \in Q$ be two of its states, and M be the maximum size of limit transitions in $\mathcal{A}$, i.e., $M = \max\{|P| \mid \exists q \in Q. q \rightarrow P \in \Delta^\ell \vee P \rightarrow q \in \Delta^\ell\}$.

We consider the problem of deciding reachability between q_I and q_F over $\mathbb{R}$, i.e., deciding whether there exists a path in $\mathcal{A}$ that starts in q_I , ends in q_F , and is labeled by a word indexed by $\mathbb{R}$. Following an analogous approach as in [29], this algorithm first generates a sequence of finite-word automata $\mathcal{A}_0, \dots, \mathcal{A}_M$ sharing the same set of states as $\mathcal{A}$. Each automaton $\mathcal{A}_j$ is such that the existence of a path in $\mathcal{A}_j$ (obeying specific restrictions that we will characterize) from a state q to a state q' is equivalent to the existence of a j -limited path from q to q' in $\mathcal{A}$ that is labeled by a word indexed by $\mathbb{R}$. Recall that each limit transition $P \rightarrow q$ and $q \rightarrow P$ in a j -limited path is such that $|P| \leq j$. Therefore all paths are systematically M -limited. Once the last automaton of the sequence has been constructed, it only remains to test whether there exists a path (of the appropriate form) from q_I to q_F in $\mathcal{A}_M$.

For every $j \in [0, M]$, the automaton $\mathcal{A}_j$ (conveniently represented here as an automaton on linear orderings without any limit transition) is of the form $(Q, 2^Q, I, F, \Delta_j, \emptyset)$, i.e., the successor transitions in Δ_j (which we define later) are labeled by subsets of Q . We introduce the following useful definitions. An *open $\mathbb{R}$ -path* $\bar{r}$ from q_1 to q'_n in $\mathcal{A}_j$ labeled by $P = P_1 \cup \dots \cup P_n$ is a finite sequence of transitions $(q_1, P_1, q'_1), \dots, (q_n, P_n, q'_n) \in \Delta_j$ such that, for every $i \in [1, n-1]$, $(q'_i, \alpha_i, q_{i+1}) \in \Delta^s$ for some $\alpha_i \in \Sigma$. Intuitively, such a path corresponds to a finite concatenation of open intervals of $\mathbb{R}$ (corresponding to transitions in $\mathcal{A}_j$), connected by single elements (corresponding to successor transitions in $\mathcal{A}$), resulting in an open interval of $\mathbb{R}$. Analogously, a *closed $\mathbb{R}$ -path* $\bar{r}$ from p to p' in $\mathcal{A}_j$ labeled by $P \cup \{p, p'\}$ is a triple consisting of an open $\mathbb{R}$ -path from a state q_1 to a state q'_n (labeled by P) together with two transitions $(p, \alpha, q_1), (q'_n, \alpha', p') \in \Delta^s$, for some $\alpha, \alpha' \in \Sigma$.

The transition relation Δ_j is defined by induction over j , with $\Delta_0 = \emptyset$, and, for $0 < j \leq M$, $\Delta_j = \Delta_{j-1} \cup S_j \cup L_j \cup R_j$, where S_j , L_j , and R_j are sets of transitions constructed from $\mathcal{A}_{j-1}$ by the following procedure. The sets S_j , L_j , and R_j are initially empty and filled according to the shuffle, infinite repetition, and reverse infinite repetition rules developed in the next sections. The final

step of the algorithm consists in searching whether there exists an open $\mathbb{R}$ -path in $\mathcal{A}_M$ that starts in q_I and ends in q_F . In the positive case, $\mathcal{A}$ admits a path labeled by a word indexed by $\mathbb{R}$ from q_I to q_F , otherwise it does not.

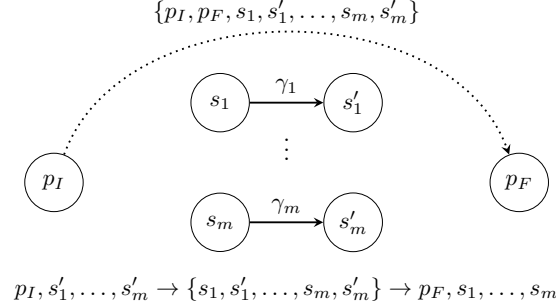


Figure 5.2: Simple shuffle rule.

Shuffle rule For every pair of transitions $p_I \rightarrow P \in \Delta^\ell$ and $P \rightarrow p_F \in \Delta^\ell$ sharing a common set P such that $|P| = j$, the transition $(p_I, P \cup \{p_I, p_F\}, p_F)$ is added to S_j if and only if there exist

- $m > 0$ successor transitions $(s_1, \gamma_1, s'_1), \dots, (s_m, \gamma_m, s'_m)$ in Δ^s , such that, for each $i \in [1, m]$, $s_i, s'_i \in P$, $P \rightarrow s_i \in \Delta^\ell$, and $s'_i \rightarrow P \in \Delta^\ell$, and
- $n \geq 0$ closed $\mathbb{R}$ -paths $\bar{r}_1, \dots, \bar{r}_n$ in $\mathcal{A}_{j-1}$, where each $\bar{r}_i$ is labeled by a set of states Q_i , starts in a state q_i such that $P \rightarrow q_i \in \Delta^\ell$, and ends in a state q'_i such that $q'_i \rightarrow P \in \Delta^\ell$

satisfying the following condition:

$$\bigcup_{i \in [1, m]} \{s_i, s'_i\} \cup \bigcup_{i \in [1, n]} Q_i = P.$$

Intuitively, there are two kinds of shuffles. A *simple shuffle* involves no closed $\mathbb{R}$ -path (corresponding to the case $n = 0$ in the shuffle rule) and is a dense combination of single elements only. The simple shuffle rule is depicted in Figure 5.2. The solid transitions correspond to successor transitions in $\mathcal{A}$, and the dotted transition shows the new transition introduced in $\mathcal{A}_j$. A *Cantor shuffle* is a dense combination of intervals of order type $1 + \lambda + 1$ (corresponding to closed $\mathbb{R}$ -paths in $\mathcal{A}_{j-1}$) and single elements (corresponding to successor transitions in $\mathcal{A}$). The Cantor shuffle rule is depicted in Figure 5.3. The dashed transitions correspond to preexisting transition in the automaton $\mathcal{A}_{j-1}$.

As a side note, consider again the automaton $\mathcal{A}_{>\mathbb{R}}$ mentioned in Lemma 5.2.1. The requirement $m > 0$ prevents the shuffle rule to be applicable with $p_I = q_1$ and $p_F = q_8$, in accordance with the fact that the automaton does not admit a path from q_1 to q_8 labeled by a word indexed by $\mathbb{R}$.

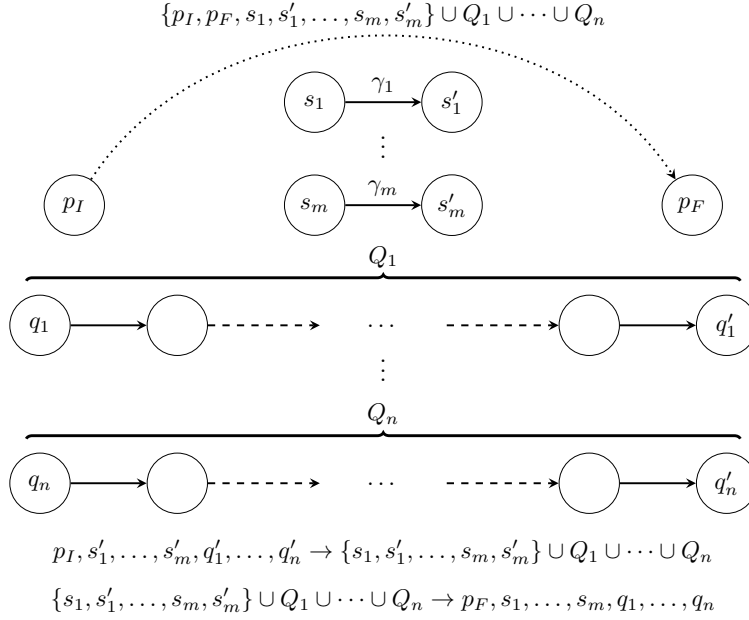


Figure 5.3: Cantor shuffle rule.

Infinite repetition rule For every transition $P \rightarrow q \in \Delta^\ell$ such that $|P| = j$, and for every state $s \in P$, the transition $(s, P \cup \{q\}, q)$ is added to L_j if and only if there exists $s' \in P$ satisfying the two following conditions:

- $\mathcal{A}_{j-1}$ admits an open $\mathbb{R}$ -path from s to s' labeled by P of the form $(s_1, P_1, s'_1), \dots, (s_n, P_n, s'_n)$ with $s = s_1$ and $s' = s'_n$.
- $\mathcal{A}$ admits a successor transition of the form $(s', \alpha, s) \in \Delta^s$.

Intuitively, this rule looks for a cycle that alternates between open intervals of order type λ (corresponding to open $\mathbb{R}$ -paths in $\mathcal{A}_{j-1}$) and single elements (corresponding to successor transitions in $\mathcal{A}$), such that the set of states visited by the whole cycle is exactly P . If such a cycle exists, then it can be repeated infinitely many times, therefore allowing to follow the limit transition $P \rightarrow q$. Notice that the starting point $s = s_1$ of such a cycle can be chosen arbitrarily, hence the rule can also be applied for every state s_i visited by the cycle, for $i \in [1, n]$, as illustrated in Figure 5.4.

Reverse infinite repetition rule This rule follows a similar principle as the infinite repetition rule, but for a reversed ordering. For every transition $q \rightarrow P \in \Delta^\ell$ such that $|P| = j$, and for every state $s' \in P$, the transition $(q, P \cup \{q\}, s')$ is added to R_j if and only if there exists $s \in P$ satisfying the two following conditions:

- $\mathcal{A}_{j-1}$ admits an open $\mathbb{R}$ -path from s to s' labeled by P of the form $(s_1, P_1, s'_1), \dots, (s_n, P_n, s'_n)$ with $s = s_1$ and $s' = s'_n$.
- $\mathcal{A}$ admits a successor transition of the form $(s', \alpha, s) \in \Delta^s$.

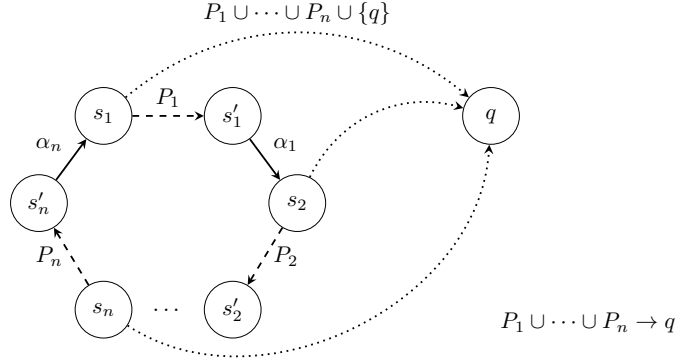


Figure 5.4: Infinite repetition rule.

5.2.3 Correctness

We state in the two following theorems that the algorithm described in Section 5.2.2 is sound and complete.

Theorem 5.2.2 (Soundness). *If there exists an open $\mathbb{R}$ -path in $\mathcal{A}_M$ from q_I to q_F labeled by P , then there exists a path in $\mathcal{A}$ from q_I to q_F , labeled by a word indexed by $\mathbb{R}$, that visits exactly the states in P .*

Theorem 5.2.3 (Completeness). *If the automaton $\mathcal{A}$ admits a path from q_I to q_F , labeled by a word indexed by $\mathbb{R}$, that visits a set of states P , then there exists an open $\mathbb{R}$ -path in $\mathcal{A}_M$ from q_I to q_F labeled by P .*

The proofs of these theorems are given in Section 5.4.

5.2.4 Example

We illustrate our algorithm, and more specifically the construction of the automata $\mathcal{A}_0, \dots, \mathcal{A}_M$ with a simple example. Consider the automaton $\mathcal{A}$ depicted in Figure 5.5. The automaton $\mathcal{A}_0$ comprises the states of automaton $\mathcal{A}$ without any transition. Since $\mathcal{A}$ does not admit a limit transition of size smaller than 2, automata $\mathcal{A}_2$ and $\mathcal{A}_1$ are identical to $\mathcal{A}_0$. The automaton $\mathcal{A}_3$ shown in Figure 5.6 has two transitions $(q_4, \{q_3, q_4, q_5\}, q_3)$ and $(q_5, \{q_3, q_4, q_5\}, q_3)$ resulting from the application of the simple shuffle rule on the set $\{q_3, q_4, q_5\}$ with $p_F = q_3$, and respectively $p_I = q_4$ and $p_I = q_5$. Since $\mathcal{A}$ does not admit a limit transition of size 4, $\mathcal{A}_4$ is identical to $\mathcal{A}_3$. The final automaton $\mathcal{A}_5$ is illustrated

in Figure 5.7. The new transitions result from the application of the shuffle rule on the set $\{q_1, q_2, q_3, q_4, q_5\}$, for every $p_I \in \{q_1, q_4, q_5\}$ and $p_F \in \{q_2, q_3\}$.

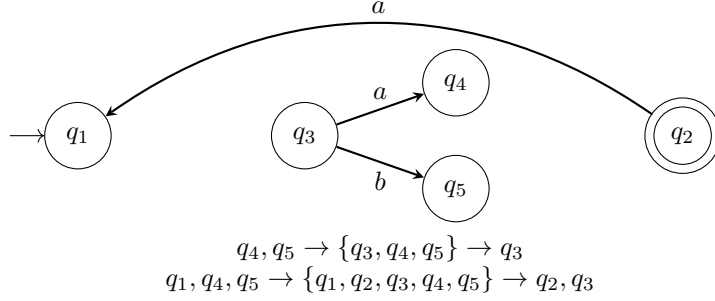


Figure 5.5: The input automaton $\mathcal{A}$.

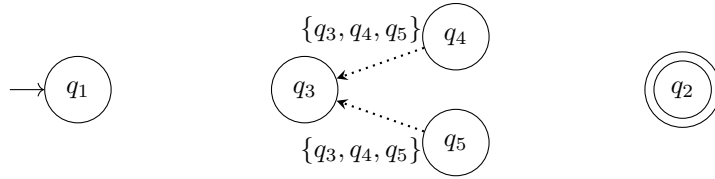


Figure 5.6: The finite-word automata $\mathcal{A}_3$ and $\mathcal{A}_4$.

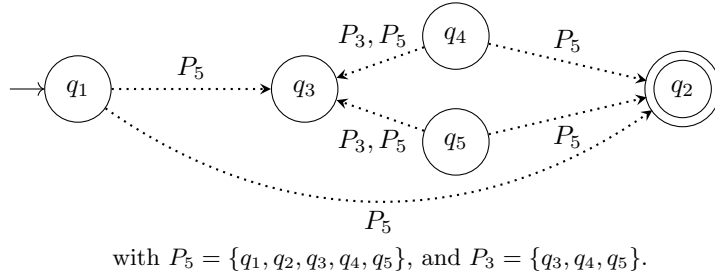


Figure 5.7: The finite-word automaton $\mathcal{A}_5$.

5.2.5 Complexity

The algorithm described in Section 5.2.2 runs in polynomial time w.r.t. the size of $\mathcal{A}$, i.e., $|Q| + |\Delta^s| + |\Delta^\ell|$.

Theorem 5.2.4. *The automaton $\mathcal{A}_M$ can be computed in polynomial time in the size of $\mathcal{A}$, by systematic application of the shuffle, infinite repetition and reverse infinite repetition rules.*

Proof. First, for each $j \in [0, M]$, each rule is called only polynomially many times during the construction of the automaton $\mathcal{A}_j$. Indeed, the shuffle rule is only considered for every combination of two limit transitions in $\mathcal{A}$, and the (reverse) infinite repetition rule is only considered for every pair consisting of a limit transition in $\mathcal{A}$ together with a state of $\mathcal{A}$.

It remains to show that each rule can be applied in polynomial time. The case of the simple shuffle rule is immediate. The other rules require to be able to check either the existence of an open $\mathbb{R}$ -path labeled by a given set of states P (to deal with the infinite and reverse infinite repetition rules), or the existence of a set of closed $\mathbb{R}$ -paths and successor transitions such that their combined labels and visited states are equal to P (to deal with the Cantor shuffle rule). These two operations can be carried out by a polynomial procedure that, given a set of states P , two states $q, q' \in P$, and the automaton $\mathcal{A}_j$, computes the set $S_{\text{open}}(\mathcal{A}_j, P, q, q') = \{r \in R \mid R \text{ labels an } \mathbb{R}\text{-path in } \bar{R} \wedge R \subseteq P\}$ where $\bar{R}$ is the set of open $\mathbb{R}$ -paths from q to q' in $\mathcal{A}_j$, and similarly for $S_{\text{closed}}(\mathcal{A}_j, P, q, q')$. This procedure works as follows. Let $\delta_{P,j,q}^{\rightarrow} \subseteq \Delta_j$ (resp. $\delta_{P,j,q'}^{\leftarrow}$) be the set of transitions occurring in (open or closed) $\mathbb{R}$ -paths of $\mathcal{A}_j$ starting in q (resp. ending in q'), and labeled by a subset of P . The sets $\delta_{P,j,q}^{\rightarrow}$ and $\delta_{P,j,q'}^{\leftarrow}$ can be computed in polynomial time by performing a (respectively forward and backward) traversal of the automaton $\mathcal{A}_j$ that respects the structure of $\mathbb{R}$ -paths, i.e., alternating between transitions in Δ_j and transitions in Δ^s . The set $S_{\text{open}}(\mathcal{A}_j, P, q, q')$ (resp. $S_{\text{closed}}(\mathcal{A}_j, P, q, q')$) is then obtained by taking the union of the labels of the transitions in $\delta_{P,j,q}^{\rightarrow} \cap \delta_{P,j,q'}^{\leftarrow}$.

For the (reverse) infinite repetition rule, consider a set of states P , and a transition $(p', \alpha, p) \in \Delta^s$ with $p, p' \in P$. If $S_{\text{open}}(\mathcal{A}_j, P, p, p') = P$, there necessarily exists an open $\mathbb{R}$ -path from p to p' in $\mathcal{A}_j$ labeled by P . Such a path can be constructed by first choosing, for each $q \in P$, an open $\mathbb{R}$ -path $\bar{r}_q$ labeled by $R \subseteq P$ such that $q \in R$, and then concatenating these paths with the help of the transition $(p', \alpha, p) \in \Delta^s$.

Finally, for the shuffle rule, consider a set of states P involved in a limit transition. We define $L_P = \{p \in P \mid P \rightarrow p \in \Delta^\ell\}$, and $R_P = \{p \in P \mid p \rightarrow P \in \Delta^\ell\}$. Let the set of states $S \subseteq P$ correspond to the states $s_1, s'_1, \dots, s_m, s'_m$ involved in the successor transitions satisfying the shuffle rule constraints. Then, using a similar argument as before, one shows that checking the existence of a set of closed $\mathbb{R}$ -paths $\bar{r}_1, \dots, \bar{r}_n$, respectively labeled by $Q_1, \dots, Q_n$, starting from a state in L_P , and ending in a state in R_P , such that $P = S \cup \bigcup_{i \in [1, n]} Q_i$, reduces to checking that $P = S \cup \bigcup_{(q, q') \in L_P \times R_P} S_{\text{closed}}(\mathcal{A}_j, P, q, q')$. $\square$

5.2.6 Pseudo Code

We now describe in pseudo-code the algorithm computing the set of transitions Δ_j of the automaton $\mathcal{A}_j$. Recall that the transitions in Δ_j must account for every j -limited path labeled by a word indexed by $\mathbb{R}$ in the input automaton $\mathcal{A} = (Q, \Sigma, I, F, \Delta^s, \Delta^\ell)$. The set of transitions Δ_j is computed recursively, with $\Delta_0 = \emptyset$, and Δ_{j-1} being used in the computation of Δ_j for $j > 0$. The procedure `Compute- $\Delta_j$` described in Algorithm 1 computes the set of transitions

Δ_j given $\mathcal{A}$ and Δ_{j-1} . This procedure relies on the subroutine **Labels- $\mathbb{R}$ -path** described in Algorithm 2, that works as follows.

Algorithm 1: Compute- Δ_j

Input: $\Delta_{j-1}, \Delta^s, \Delta^\ell$
Output: The set of transitions Δ_j corresponding to the automaton $\mathcal{A}_j$.

```

1  $L_j, R_j, S_j \leftarrow \emptyset$ 
2 foreach  $P \subseteq Q$  such that  $|P| = j \wedge \exists q \in Q. (q \rightarrow P) \in \Delta^\ell \vee (P \rightarrow q) \in \Delta^\ell$  do
3   foreach  $(p', \alpha, p) \in \Delta^s$  such that  $p, p' \in P$  do
4     if Labels- $\mathbb{R}$ -path $(\Delta_{j-1}, \Delta^s, P, p, p', \text{open}) = P$  then
5       foreach  $q \in Q$  such that  $(P \rightarrow q) \in \Delta^\ell$  do
6          $L_j \leftarrow L_j \cup \{(p, P \cup \{q\}, q)\}$  /* Infinite rep. */
7       foreach  $q \in Q$  such that  $(q \rightarrow P) \in \Delta^\ell$  do
8          $R_j \leftarrow R_j \cup \{(q, P \cup \{q\}, p')\}$  /* Rev. infinite rep. */
9    $\text{Left}_P \leftarrow \{q \in Q \mid (P \rightarrow q) \in \Delta^\ell\}$  /* Shuffles */
10   $\text{Right}_P \leftarrow \{q \in Q \mid (q \rightarrow P) \in \Delta^\ell\}$ 
11   $\text{atLeastOneIndividualPoint} \leftarrow \text{false}$ 
12   $\text{StatesVisited} \leftarrow \emptyset$ 
13  foreach  $(\ell, r) \in (\text{Left}_P \cap P) \times (\text{Right}_P \cap P)$  do
14    foreach  $(\ell, \alpha, r) \in \Delta^s$  do
15       $\text{atLeastOneIndividualPoint} \leftarrow \text{true}$ 
16       $\text{StatesVisited} \leftarrow \text{StatesVisited} \cup \{\ell, r\}$ 
17      break
18   $\text{StatesVisited} \leftarrow \text{StatesVisited} \cup \text{Labels-}\mathbb{R}\text{-path}(\Delta_{j-1}, \Delta^s, P, \ell, r, \text{closed})$ 
19  if  $\text{StatesVisited} = P \wedge \text{atLeastOneIndividualPoint}$  then
20    foreach  $(p_I, p_F) \in \text{Right}_P \times \text{Left}_P$  do
21       $S_j \leftarrow S_j \cup \{(p_I, P \cup \{p_I, p_F\}, p_F)\}$ 
22  $\Delta_j \leftarrow \Delta_{j-1} \cup L_j \cup R_j \cup S_j$ 

```

For a given automaton $\mathcal{A}_j$, a set of states $P \subseteq Q$, and two states $q, q' \in P$, the subroutine **Labels- $\mathbb{R}$ -path** computes either the set $S_{\text{open}}(\mathcal{A}_j, P, q, q')$, or the set $S_{\text{closed}}(\mathcal{A}_j, P, q, q')$ mentioned in Section 5.2.5, depending on the value of the **type** parameter. It first performs a search (or traversal) in the automaton $\mathcal{A}_j$ starting from the state q , and collects every transition in Δ_j that allows to progress in the search by following an $\mathbb{R}$ -path labeled by subsets of P . Once this search is over, it performs a backward search in $\mathcal{A}_j$ starting from q' , that collects every transitions allowing to progress in this traversal by following an $\mathbb{R}$ -path labeled by a subset of P . In this backward search, we actually only consider the transitions in $\mathcal{A}_j$ that have been collected during the first forward search. Once this second search is done, we return the union of the labels of every transition collected during the backward search, which by construction was first collected during the forward search, hence belonging to an $\mathbb{R}$ -path from q to q' in $\mathcal{A}_j$. The procedure **Labels- $\mathbb{R}$ -path** runs in polynomial time.

Algorithm 2: Labels- $\mathbb{R}$ -path

Input: $\Delta_{j-1}, \Delta^s, P \subseteq Q, q \in P, q' \in P, \text{type} \in \{\text{open}, \text{closed}\}$.
Output: The set of states (subset of P) occurring in labels of *open* or *closed* (depending on the *type* parameter) $\mathbb{R}$ -paths from q to q' , in $\mathcal{A}_{j-1}$.

```

1   $E_I \leftarrow \emptyset$  /* Transitions from  $\Delta_{j-1}$  reachable from  $q$  */
2   $E_P \leftarrow \emptyset$  /* Transitions from  $\Delta^s$  reachable from  $q$  */
3   $\text{VisitedI} \leftarrow \emptyset$  /* States already visited when looking for an interval */
4   $\text{VisitedP} \leftarrow \emptyset$  /* States already visited when looking for a point */
5  if type = open then
6     $(\text{ToCheckI}, \text{ToCheckP}) \leftarrow (\{q\}, \emptyset)$ 
7  else
8     $(\text{ToCheckI}, \text{ToCheckP}) \leftarrow (\emptyset, \{q\})$ 
9  while  $(\text{ToCheckI}, \text{ToCheckP}) \neq (\emptyset, \emptyset)$  do
10   foreach  $p \in \text{ToCheckI}$  do
11      $(\text{VisitedI}, \text{ToCheckI}) \leftarrow (\text{VisitedI} \cup \{p\}, \text{ToCheckI} \setminus \{p\})$ 
12     foreach  $(p, R, r) \in \Delta_{j-1}$  such that  $R \subseteq P$  do
13        $E_I \leftarrow E_I \cup \{(p, R, r)\}$ 
14       if  $r \notin \text{VisitedP}$  then  $\text{ToCheckP} \leftarrow \text{ToCheckP} \cup \{r\}$ 
15   foreach  $p \in \text{ToCheckP}$  do
16      $(\text{VisitedP}, \text{ToCheckP}) \leftarrow (\text{VisitedP} \cup \{p\}, \text{ToCheckP} \setminus \{p\})$ 
17     foreach  $(p, \alpha, r) \in \Delta^s$  such that  $\{p, r\} \subseteq P$  do
18        $E_P \leftarrow E_P \cup \{(p, \alpha, r)\}$ 
19       if  $r \notin \text{VisitedI}$  then  $\text{ToCheckI} \leftarrow \text{ToCheckI} \cup \{r\}$ 

/* We only keep the transitions in  $E_I \cup E_P$  that allow to reach  $q'$ , */
/* by performing the same search but backwards, starting from  $q'$ . */
20  $S \leftarrow \emptyset$ 
21  $(\text{VisitedIBack}, \text{VisitedPBack}) \leftarrow (\emptyset, \emptyset)$ 
22 if type = open then
23    $(\text{ToCheckIBack}, \text{ToCheckPBack}) \leftarrow (\{q'\}, \emptyset)$ 
24 else
25    $(\text{ToCheckIBack}, \text{ToCheckPBack}) \leftarrow (\emptyset, \{q'\})$ 
26 while  $(\text{ToCheckIBack}, \text{ToCheckPBack}) \neq (\emptyset, \emptyset)$  do
27   foreach  $p \in \text{ToCheckIBack}$  do
28      $(\text{VisitedIBack}, \text{ToCheckIBack}) \leftarrow (\text{VisitedIBack} \cup \{p\}, \text{ToCheckIBack} \setminus \{p\})$ 
29     foreach  $(r, R, p) \in E_I$  do /*  $p$  is the endpoint */
30        $S \leftarrow S \cup R$ 
31       if  $r \notin \text{VisitedPBack}$  then  $\text{ToCheckPBack} \leftarrow \text{ToCheckPBack} \cup \{r\}$ 
32   foreach  $p \in \text{ToCheckPBack}$  do
33      $(\text{VisitedPBack}, \text{ToCheckPBack}) \leftarrow (\text{VisitedPBack} \cup \{p\}, \text{ToCheckPBack} \setminus \{p\})$ 
34     foreach  $(r, \alpha, p) \in E_P$  do /*  $p$  is the endpoint */
35        $S \leftarrow S \cup \{r, p\}$ 
36       if  $r \notin \text{VisitedIBack}$  then  $\text{ToCheckIBack} \leftarrow \text{ToCheckIBack} \cup \{r\}$ 
37 return  $S$ 

```

5.3 Reachability over the Rationals

The reachability algorithm over the reals can be adapted to accommodate reachability over the rational numbers $\mathbb{Q}$. Because of the incompleteness of $\mathbb{Q}$, the rules must be adapted in order to account for the presence of gaps.

5.3.1 Algorithm

We first define a notion of valid path over $\mathbb{Q}$, analogous to the notion of $\mathbb{R}$ -path. An open $\mathbb{Q}$ -path from q_1 to q'_n in $\mathcal{A}_j$ labeled by $P = P_1 \cup \dots \cup P_n$ is a finite sequence of transitions $(q_1, P_1, q'_1), \dots, (q_n, P_n, q'_n) \in \Delta_j$ such that for each $i \in [1, n-1]$, either $q'_i = q_{i+1}$, or there exists a successor transition $(q'_i, \alpha_i, q_{i+1}) \in \Delta^s$. Intuitively, such a path corresponds to a finite concatenation of open intervals of $\mathbb{Q}$ (corresponding to transitions in $\mathcal{A}_j$), in which two consecutive intervals are either connected by a gap (if the shared bound between the two intervals is an irrational number in $\mathbb{R} \setminus \mathbb{Q}$, e.g., $\sqrt{2}$), or by a single element (corresponding to a successor transition in $\mathcal{A}$, if the shared bound belongs to $\mathbb{Q}$). Like in the case of $\mathbb{R}$, we also define *closed* $\mathbb{Q}$ -paths, i.e., open $\mathbb{Q}$ -paths preceded and followed by a successor transition so as to form a closed interval, as well as *half-open* $\mathbb{Q}$ -paths, i.e., open $\mathbb{Q}$ -paths preceded or (exclusively) followed by one successor transition.

For a given set of states P involved in a limit transition, both the simple and Cantor shuffle rules additionally require a non-empty set of states $(g_1, \dots, g_k)$ such that $g_i \rightarrow P \rightarrow g_i$, for all $i \in [1, k]$, corresponding to gaps. Also, the left and right infinite repetition rules must be adapted to allow for the required $\mathbb{Q}$ -path to be cyclic, i.e., $q'_n = q_1$. We now describe these rules formally.

Shuffle rule For every pair of transitions $p_I \rightarrow P \in \Delta^\ell$ and $P \rightarrow p_F \in \Delta^\ell$ sharing a common set P such that $|P| = j$, the transition $(p_I, P \cup \{p_I, p_F\}, p_F)$ is added to S_j if and only if there exist

- $m \geq 0$ successor transitions $(s_1, \gamma_1, s'_1), \dots, (s_m, \gamma_m, s'_m)$ in Δ^s , such that, for each $i \in [1, m]$, $s_i, s'_i \in P$, $P \rightarrow s_i \in \Delta^\ell$, and $s'_i \rightarrow P \in \Delta^\ell$,
- $n \geq 0$ open, closed, or half-open $\mathbb{Q}$ -paths $\bar{q}_1, \dots, \bar{q}_n$ in $\mathcal{A}_{j-1}$, where each $\bar{q}_i$ is labeled by a set of states Q_i , starts in a state q_i such that $P \rightarrow q_i \in \Delta^\ell$, and ends in a state q'_i such that $q'_i \rightarrow P \in \Delta^\ell$, and
- $k > 0$ states $g_1, \dots, g_k$ in P such that, for every $i \in [1, k]$, both transitions $g_i \rightarrow P$ and $P \rightarrow g_i$ belong to Δ^ℓ

satisfying the two following conditions:

- $\bigcup_{i \in [1, m]} \{s_i, s'_i\} \cup \bigcup_{i \in [1, k]} \{g_i\} \cup \bigcup_{i \in [1, n]} Q_i = P$.
- The inequality $n + m > 0$ holds (otherwise, there would only be gaps).

Infinite repetition rule For every transition $P \rightarrow q \in \Delta^\ell$ such that $|P| = j$, and for every state $s \in P$, the transition $(s, P \cup \{q\}, q)$ is added to L_j if and only if there exists $s' \in P$ satisfying the two following conditions:

- $\mathcal{A}_{j-1}$ admits an open $\mathbb{Q}$ -path from s to s' labeled by P of the form $(q_1, P_1, q'_1), \dots, (q_n, P_n, q'_n)$ with $s = q_1$ and $s' = q'_n$.
- Either $s = s'$, or $\mathcal{A}$ admits a successor transition of the form $(s', \alpha, s) \in \Delta^s$.

Reverse infinite repetition rule For every transition $q \rightarrow P \in \Delta^\ell$ such that $|P| = j$, and for every state $s' \in P$, the transition $(q, P \cup \{q\}, s')$ is added to R_j if and only if there exists $s \in P$ satisfying the two following conditions:

- $\mathcal{A}_{j-1}$ admits an open $\mathbb{Q}$ -path from s to s' labeled by P of the form $(q_1, P_1, q'_1), \dots, (q_n, P_n, q'_n)$ with $s = q_1$ and $s' = q'_n$.
- Either $s = s'$, or $\mathcal{A}$ admits a successor transition of the form $(s', \alpha, s) \in \Delta^s$.

5.3.2 Example

First, observe that running the algorithm for $\mathbb{Q}$ over the automaton $\mathcal{A}$ (depicted in Figure 5.5 and studied in Section 5.2.4) does not generate a single new transition. This is because the automaton $\mathcal{A}$ lacks states that can be matched with gaps, which implies that no shuffle rule over $\mathbb{Q}$ can be applied.

Now, let us illustrate our algorithm for $\mathbb{Q}$ over the automaton $\mathcal{B}$ depicted in Figure 5.8. The automaton $\mathcal{B}_2$ shown in Figure 5.9a has four transitions. The two transitions $(q_2, \{q_1, q_2\}, q_1)$ and $(q_2, \{q_1, q_2\}, q_2)$ result from the application of the shuffle rule on the set $\{q_1, q_2\}$ with $p_I = q_2$, and respectively $p_F = q_1$ and $p_F = q_2$. Notice that in this application of the shuffle rule, the state q_2 behaves both as the endpoint of a successor transition, and as a state that can be matched to a gap. The two transitions $(q_1, \{q_1, q_3\}, q_1)$ and $(q_1, \{q_1, q_3\}, q_3)$ result from a similar application of the shuffle rule on the set $\{q_1, q_3\}$ with $p_I = q_1$, and respectively $p_F = q_1$ and $p_F = q_3$. This time, it is q_1 that acts both as the endpoint of a successor transition, and as a state that can be matched to a gap.

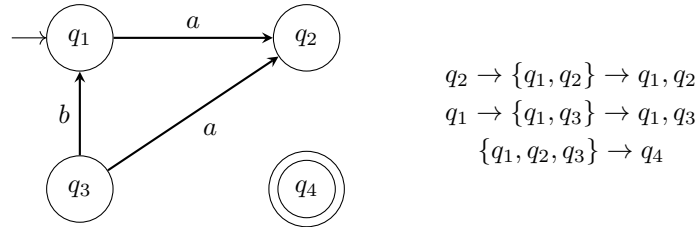


Figure 5.8: The input automaton $\mathcal{B}$.

The automaton $\mathcal{B}_3$ shown in Figure 5.9b has two additional transitions. The transition $(q_1, \{q_1, q_2, q_3, q_4\}, q_4)$ results from the application of the infinite repetition rule on the set $\{q_1, q_2, q_3\}$, with $s = s' = q_1$, which is valid thanks to the existence of the open $\mathbb{Q}$ -path from q_1 to itself $q_1 \xrightarrow{\{q_1, q_3\}} q_3 \xrightarrow{a} q_2 \xrightarrow{\{q_1, q_2\}} q_1$. The transition $(q_2, \{q_1, q_2, q_3, q_4\}, q_4)$ results from the application of the infinite repetition rule on the set $\{q_1, q_2, q_3\}$, with $s = q_2$ and $s' = q_3$, using the open $\mathbb{Q}$ -path $q_2 \xrightarrow{\{q_1, q_2\}} q_1 \xrightarrow{\{q_1, q_3\}} q_3$, and the successor transition (q_3, a, q_2) .

Notice that the application of the infinite repetition rule does not generate the transition $(q_3, \{q_1, q_2, q_3, q_4\}, q_4)$. This is because it is not possible to find

an open $\mathbb{Q}$ -path that starts in q_3 , i.e., there does not exist a transition in $\mathcal{B}_2$ that originates from q_3 .

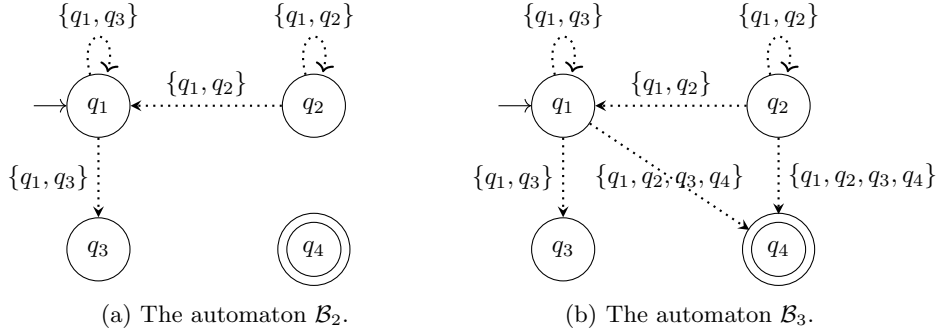


Figure 5.9: The automata obtained at the main steps of the algorithm for $\mathbb{Q}$.

5.4 Proof of Correctness

We now prove the correctness of the algorithm introduced in Section 5.2.2.

5.4.1 Soundness

In order to prove the soundness of the algorithm (Theorem 5.2.2), we first show (Lemma 5.4.1) that an open $\mathbb{R}$ -path $\bar{r}$ in $\mathcal{A}_M$ corresponds to a path in $\mathcal{A}$ indexed by $\mathbb{R}$, provided that each transition of $\mathcal{A}_M$ involved in $\bar{r}$ corresponds to a path in $\mathcal{A}$ indexed by $\mathbb{R}$. Then, it suffices to prove (Lemma 5.4.3) that each transition in $\mathcal{A}_M$ corresponds to a path in $\mathcal{A}$ indexed by $\mathbb{R}$.

Lemma 5.4.1. *Let $(q_1, P_1, q'_1), \dots, (q_n, P_n, q'_n)$ be an open $\mathbb{R}$ -path in $\mathcal{A}_M$, such that for every $i \in [1, n]$, there exists a path $q_i \rightarrow q'_i$ in $\mathcal{A}$ labeled by a word indexed by $\mathbb{R}$, visiting the set of states P_i . Then, there exists a path $q_1 \rightarrow q'_n$ in $\mathcal{A}$ labeled by a word indexed by $\mathbb{R}$, that visits exactly the set of states $P_1 \cup \dots \cup P_n$.*

Proof. The proof relies on the fact that a path labeled by a word indexed by $\mathbb{R}$ can also be labeled by a word indexed by any non-empty open interval $(x, y) \subseteq \mathbb{R}$, since both share the order type λ .

We consider a sequence of words $w_1, w_2, \dots, w_n$ such that:

- w_1 is indexed by $(-\infty, 1)$ and w_n is indexed by $(n-1, +\infty)$;
- for all $i \in [2, n-1]$, w_i is indexed by $(i-1, i)$;
- for all $i \in [1, n]$, w_i labels a path π_i in $\mathcal{A}$ from q_i to q'_i .

Recall that for $(q_1, P_1, q'_1), \dots, (q_n, P_n, q'_n)$ to be an open $\mathbb{R}$ -path, there must exist a successor transition of the form $(q'_i, \alpha_i, q_{i+1}) \in \Delta^s$, for every $i \in [1, n-1]$. In order to construct a word indexed by $\mathbb{R}$ that labels a path from q_1 to q'_n , we

simply concatenate together the words w_i , using these successor transitions to fill the gaps between consecutive open intervals. The paths π_i can be combined into a path π from q_1 to q'_n , labeled by the word $w = w_1 \cdot \alpha_1 \cdot w_2 \cdot \dots \cdot w_{n-1} \cdot \alpha_{n-1} \cdot w_n$. By construction, the path π is indexed by $\mathbb{R}$. $\square$

Before advancing, we first introduce the following result which will be needed in the remainder of the proof of Theorem 5.2.2.

Lemma 5.4.2. *Let $m > 0$. There exists a partition of $\mathbb{R}$ into m non-empty pairwise-disjoint subsets $R_1, \dots, R_m$ such that for every $i \in [1, m]$, R_i is dense in $\mathbb{R}$.*

Proof. Let us construct such a partition of $\mathbb{R}$. We define, for every $k \in \mathbb{N}_{>1}$, the set S_k denoting the set of (non-integer) rational numbers that can be represented by a fraction with a strictly positive power of k as its denominator. Formally, $S_k = \{p/k^j \mid p \in \mathbb{Z} \text{ and } j \in \mathbb{N}_{>0}\} \setminus \mathbb{Z}$.

Let us show that, for any $k > 1$, the set S_k is dense in $\mathbb{R}$. We simply need to show that within any non-empty open interval $(x, y) \subseteq \mathbb{R}$, there exists $z \in S_k$ such that $x < z < y$. Since $k > 1$, we know that there exists $j \in \mathbb{N}$ such that $1/k^j < (y - x)/2$. Let us pick such a value j . There also exists an integer $p \in \mathbb{Z}$ such that $p/k^j \leq (x + y)/2 \leq (p + 1)/k^j$, one could pick $p = \lfloor (x + y) \cdot (k^j)/2 \rfloor$ for instance. Notice that $(x + y)/2$ is the middle of the interval (x, y) . Since the difference between p/k^j and $(p + 1)/k^j$ is equal to $1/k^j$ and since $1/k^j < (y - x)/2$, we have that either $p/k^j \in (x, y)$ or $(p + 1)/k^j \in (x, y)$ (it could be both).

Let us now show that for two distinct prime numbers p_1 and p_2 , the sets S_{p_1} and the set S_{p_2} are disjoint. Assume that there exist $n_1/(p_1)^i \in S_{p_1}$ and $n_2/(p_2)^j \in S_{p_2}$ for some $i, j > 0$, such that $n_1/(p_1)^i = n_2/(p_2)^j$. This yields the equality $n_1 p_2^j = n_2 p_1^i$. Since p_1 and p_2 are prime numbers, this implies $n_1 = k(p_1^i)$ and $n_2 = k(p_2^j)$ for some $k \in \mathbb{Z}$. This is impossible because $k(p_1^i)/p_1^i = k \in \mathbb{Z}$, and we require $S_{p_1} \cap \mathbb{Z} = \emptyset$ (the same development applies to S_{p_2}).

Let $P = (p_i)_{i \in [1, m]}$ be the sequence of the m first prime numbers (i.e., $2, 3, 5, 7, \dots$). We then simply set $R_1 = \mathbb{R} \setminus \mathbb{Q}$, $R_i = S_{p_i}$ for $i \in [2, m - 1]$, and $R_m = \mathbb{Q} \setminus (\bigcup_{i \in [2, m - 1]} S_{p_i})$. By construction, it is clear that the sets $(R_i)_{i \in [1, m]}$ form a partition of $\mathbb{R}$. The density of each R_i for $i \in [2, m - 1]$ has already been proven. The set R_1 is also dense in $\mathbb{R}$ since it corresponds to the set of irrational numbers. Finally, the density of R_m is obtained by noticing that $S_{p_m} \subsetneq R_m$, and remembering that we already proved that S_{p_m} is dense in $\mathbb{R}$. $\square$

Lemma 5.4.3. *For each transition (p, P, q) of $\mathcal{A}_j$ for $j \in [0, M]$, there exists a path from p to q in $\mathcal{A}$, labeled by a word indexed by $\mathbb{R}$, that visits exactly the states in P .*

Proof. The proof is constructive, explicitly building a word indexed by $\mathbb{R}$ that yields a path in $\mathcal{A}$. The form of the word depends on the rule that generated the transition of $\mathcal{A}_j$.

The lemma is proved by induction over j . The base case is trivial, since $\mathcal{A}_0$ does not have transitions. Assuming that every transition of $\mathcal{A}_{j-1}$ verifies the lemma, we now prove that the lemma also holds for any transition (p, P, q) of $\mathcal{A}_j$ that does not belong to $\mathcal{A}_{j-1}$.

First, consider that (p, P, q) results from the application of the *simple shuffle* rule, i.e., the shuffle rule involving only successor transitions without any closed $\mathbb{R}$ -path. Formally, we have $p \rightarrow P \in \Delta^\ell$ and $P \rightarrow q \in \Delta^\ell$, and a non-empty set of successor transitions $\{(s_1, \gamma_1, s'_1), \dots, (s_m, \gamma_m, s'_m)\} \subseteq \Delta^s$ such that $s_i, s'_i \in P$, $P \rightarrow s_i \in \Delta^\ell$, and $s'_i \rightarrow q \in \Delta^\ell$, for all $i \in [1, m]$. Let $\{R_1, \dots, R_m\}$ be a partition of $\mathbb{R}$ into m disjoint infinite dense sets, i.e., for any two real numbers x and y such that $x < y$, there exist $z_1, \dots, z_m$ such that $x < z_i < y$ and $z_i \in R_i$, for all $i \in [1, m]$. By Lemma 5.4.2, such a partition of $\mathbb{R}$ can always be found. Now, consider the word $w = (\alpha_r)_{r \in \mathbb{R}}$ indexed by $\mathbb{R}$ such that $\alpha_r = \gamma_i$ if $r \in R_i$. To build a suitable path π in $\mathcal{A}$ from p to q reading w , it suffices to impose $\pi(-\infty) = p$ and $\pi(+\infty) = q$, where $-\infty$ and $+\infty$ denote respectively the first and last cut of $\mathbb{R}$, and furthermore require $\pi(r^-) = s_i$ and $\pi(r^+) = s'_i$ if $r \in R_i$, for each $r \in \mathbb{R}$. By construction, it is clear that π is a valid path in $\mathcal{A}$ that visits exactly the states in P .

For the three remaining rules, notice that for any $\mathbb{R}$ -path $(p_1, P_1, p'_1), \dots, (p_n, P_n, p'_n)$ in $\mathcal{A}_{j-1}$, the inductive hypothesis and Lemma 5.4.1 ensure that there exists a path $p_1 \rightarrow p'_n$ in $\mathcal{A}$, labeled by a word indexed by $\mathbb{R}$ (or any non-empty open interval of $\mathbb{R}$), that visits exactly the states in $P_1 \cup \dots \cup P_n$.

If (p, P, q) is generated by the *infinite repetition* rule, i.e., it belongs to L_j but not to Δ_{j-1} , the rule's conditions imply that there exists an open $\mathbb{R}$ -path $(p_1, P_1, p'_1), \dots, (p_n, P_n, p'_n)$ in $\mathcal{A}_{j-1}$, such that:

- $p_1 = p$,
- there exists a transition $(p'_n, \alpha, p) \in \Delta^s$,
- $P = P_1 \cup \dots \cup P_n$, and
- $(P \rightarrow q) \in \Delta^\ell$.

Consider an infinite sequence of words $(w_k)_{k \in \mathbb{N}}$, each of them labeling a valid path from p to p'_n , such that w_0 is indexed by $(-\infty, 0)$, and for every $k > 0$, w_k is indexed by $(k-1, k)$. Now consider the word $w = \prod_{k \in \mathbb{N}} (w_k \cdot \alpha) = w_0 \cdot \alpha \cdot w_1 \cdot \alpha \cdot \dots$ indexed by $\mathbb{R}$. By simply concatenating the paths labeled by the words $(w_k)_{k \in \mathbb{N}}$ (each starting in p and ending in p'_n) with the transition (p'_n, α, p) , and assigning the state q to the last cut $(+\infty)$ of $\mathbb{R}$, we obtain a valid path reading w that starts in p , ends in q , and visits exactly P .

The case of *reverse infinite repetitions* is handled symmetrically. The accepted word indexed by $\mathbb{R}$ can also be expressed as an infinite concatenation of words but in reverse, i.e., w_0 is indexed by $(0, +\infty)$, and each w_k is indexed by $-(k+1, -k)$ for $k \in \mathbb{N}_{>0}$.

Finally, the case of the Cantor shuffle can be dealt with by considering the set of intervals and the set of individual points involved in the Cantor ternary set construction [27, 41], that we will describe explicitly. For simplicity, we will

actually build a word indexed by the interval $(0, 1)$ instead of $\mathbb{R}$. A corresponding word indexed by $\mathbb{R}$ can be directly derived by applying an increasing bijective function from $(0, 1)$ to $\mathbb{R}$, e.g., $f : (0, 1) \rightarrow \mathbb{R}; x \mapsto \frac{1}{1-x} - \frac{1}{x}$.

Consider the pair of states p, q , some closed $\mathbb{R}$ -paths $\bar{r}_1, \dots, \bar{r}_n$ in $\mathcal{A}_j$ (respectively starting in $q_1, \dots, q_n$, ending in $q'_1, \dots, q'_n$, and labeled by $Q_1, \dots, Q_n$), as well as some successor transitions $(s_1, \gamma_1, s'_1), \dots, (s_m, \gamma_m, s'_m)$ in Δ^s meeting the requirements for the application of the Cantor shuffle rule generating the transition (p, P, q) , for some set of states P .

In the spirit of the Cantor ternary set construction, we first define an infinite sequence of sets of intervals $(\bar{I}_k)_{k \in \mathbb{N}}$ by induction. We start by setting $\bar{I}_0 = \{ [\frac{1}{3}, \frac{2}{3}] \}$. We then define inductively $\bar{I}_{k+1}$ as the set of closed intervals that correspond to the middle third of the maximal intervals that remain from $(0, 1)$ after removing every interval in $\bar{I}_j$, for every $j \in [1, k]$. For instance, $\bar{I}_2$ is equal to $\{ [\frac{1}{9}, \frac{2}{9}], [\frac{7}{9}, \frac{8}{9}] \}$, and $\bar{I}_3$ is equal to $\{ [\frac{1}{27}, \frac{2}{27}], [\frac{7}{27}, \frac{8}{27}], [\frac{19}{27}, \frac{20}{27}], [\frac{25}{27}, \frac{26}{27}] \}$. We then define n sets of closed intervals $\bar{H}_1, \dots, \bar{H}_n$ by setting

$$\bar{H}_i = \bigcup_{\substack{k \in \mathbb{N} \\ (k=i) \bmod n}} \bar{I}_k,$$

for every $i \in [1, n]$. This definition ensures density among all these sets, i.e., for any $i \in [1, n]$, $\bar{H}_i$ is dense in $\bigcup_{j \in [1, n]} \bar{H}_j$ (where the implicit order relation between disjoint intervals is $[a, b] < [c, d]$ iff $b < c$).

For every $i \in [1, n]$ and interval $[a, b]$ in $\bar{H}_i$, we consider a word $w^{[a, b]} = (\sigma_r^{[a, b]})_{r \in [a, b]}$ indexed by $[a, b]$ that labels a path $q_i \rightarrow q'_i$ visiting the set of states Q_i . The existence of such words results from our assumption that every open $\mathbb{R}$ -path admits a corresponding word indexed by any non-empty open interval of $\mathbb{R}$, and that a closed $\mathbb{R}$ -path is simply an open $\mathbb{R}$ -path with additional successor transitions at the beginning and at the end, corresponding to the bounds of the interval.

In order to obtain a complete path over $(0, 1)$, it remains to assign a state to both the first and last cuts 0^+ and 1^- , as well as to all the individual points left aside by the Cantor ternary set construction. The first task is easy to achieve, by simply assigning the state p to 0^+ and q to 1^- . For the second task, let C be the set of points left out by the construction of the sets $\bar{H}_i$, i.e., $C = (0, 1) \setminus \bigcup_{[a, b] \in (\bar{H}_1 \cup \dots \cup \bar{H}_n)} [a, b]$. We define a partition $C_1, \dots, C_m$ of C , such that each C_j is dense in C , i.e., $\forall k \in [1, m]. \forall x, y \in C. (x < y) \Rightarrow (\exists z. x < z < y \wedge z \in C_k)$. This can be achieved, for instance, by considering the representation of each element in C in base 3. Indeed, a notable property of the Cantor ternary set construction (using closed intervals for the definition of the sequence $(\bar{I}_k)_{k \in \mathbb{N}}$) is that the set C corresponds exactly to the real numbers in $(0, 1)$ that admit a (unique) representation in base 3 of the form $0.t_1 t_2 t_3 \dots$ where $t_i \in \{0, 2\}$ for every $i \in \mathbb{N}$, that does not end with an infinite sequence of 0 or 2 digits. For $k \in [1, m-1]$, we define the set C_k as the subset of C consisting of the elements that admit a notation ending with $(0^k 2^k)^{+\omega}$, and we define $C_m = C \setminus (\bigcup_{k \in [1, m-1]} C_k)$. It is easy to show that for every $k \in [1, m]$,

C_k is dense in C . Then, we simply associate to each real number $c \in C_k$ the transition (s_k, γ_k, s'_k) , for every $k \in [1, m]$.

Let $u = (\beta_r)_{r \in (0,1)}$ be a word indexed by $(0, 1)$ defined as follows. For all $r \in C$ and $k \in [1, m]$, if $r \in C_k$ then $\beta_r = \gamma_k$. For all $r \in (0, 1) \setminus C$ and $i \in [1, n]$, if there exists $[a, b] \in \overline{H}_i$ such that $r \in [a, b]$, then $\beta_r = \sigma_r^{[a,b]}$. This defines u entirely since by construction, the sets $C_1, \dots, C_m$ together with the intervals in $\overline{H}_1, \dots, \overline{H}_n$ form a partition of $(0, 1)$. A path π reading u can then be constructed by properly assembling the paths reading each word $w^{[a,b]}$ for every $[a, b] \in \overline{H}_1 \cup \dots \cup \overline{H}_n$, and every successor transition (s_k, γ_k, s'_k) , for every $k \in [1, m]$. It follows that π is of the required form, i.e., π is a path from p to q , that reads a word indexed by $(0, 1)$, and visits exactly the states in P . $\square$

5.4.2 Completeness

The completeness of the reachability algorithm described in Section 5.2.2 (Theorem 5.2.3) is a direct corollary of the following, more general, result.

Theorem 5.4.1. *Assume that the automaton $\mathcal{A}$ admits a path from a state q_I to state q_F visiting the set of states P , labeled by a word w indexed by $\mathbb{R}$. If this path is j -limited, then there exists an open $\mathbb{R}$ -path in $\mathcal{A}_j$ from q_I to q_F labeled by P .*

The remainder of this section is dedicated to proving Theorem 5.4.1. The proof is essentially based on mechanisms introduced in [73] for establishing the decidability of the first-order theory of order. The same mechanisms have been adapted to develop a self-contained proof of decidability for $\text{MFO}(\mathbb{R}, <)$ in [13].

The proof is by induction over j . The base case $j = 0$ is trivial. Indeed, every path labeled by $\mathbb{R}$ must involve a limit transition, and limit transitions cannot involve the empty set of states. For the inductive case, let $j \in [1, M]$ be such that Theorem 5.4.1 holds for all $(j-1)$ -limited paths. Since every k -limited path with $k < j$ is also a $(j-1)$ -limited path (every transition in A_k is also a transition in A_{j-1}), it suffices to consider the case of j -limited paths that are not $(j-1)$ -limited paths, i.e., that involve at least one limit transition of size j .

Throughout the remainder of this section, we consider a j -limited path π in $\mathcal{A}$ labeled by a word $w = (\alpha_r)_{r \in \mathbb{R}}$ indexed by $\mathbb{R}$.

Let $(x, y) \subseteq \mathbb{R}$ be an non-empty open interval. We denote by $S_\pi(x, y)$ the set of states visited by π between the cuts x^+ and y^- . In the particular cases $x = -\infty$ and $y = +\infty$, x^+ and y^- respectively correspond to the first and the last cuts of $\mathbb{R}$. Formally, we have

$$S_\pi(x, y) = \{q \in Q \mid \exists z. x < z < y \wedge (\pi(z^-) = q \vee \pi(z^+) = q)\}.$$

We say that an interval (x, y) is π -stable if for every interval (a, b) with $x \leq a < b \leq y$, one has $S_\pi(a, b) = S_\pi(x, y)$. In other words, an interval is π -stable if within any subinterval, the set of states visited in the corresponding subpath of π remains the same.

We now prove that any π -stable interval corresponds to a structure satisfying the simple shuffle rule. We first show that every limit transition followed within a π -stable interval involves the same set of states, namely, the set of all the states visited by π within that interval.

Lemma 5.4.4. *Let $(x, y) \subseteq \mathbb{R}$ be a π -stable interval. The set of limit transitions followed by π over (x, y) is exactly $\{S_\pi(x, y) \rightarrow q \mid \exists x < z < y. \pi(z^-) = q\} \cup \{q \rightarrow S_\pi(x, y) \mid \exists x < z < y. \pi(z^+) = q\}$. Furthermore, π also follows the right-limit transition $\pi(x^+) \rightarrow S_\pi(x, y)$ to leave $\pi(x^+)$ at the cut x^+ , and follows the left-limit transition $S_\pi(x, y) \rightarrow \pi(y^-)$ to reach $\pi(y^-)$ at the cut y^- .*

Proof. Let us prove the left-limit transition case. The right-limit transition case is handled in a symmetrical way.

Let C_L be the set of left cuts z^- associated with the elements $z \in (x, y)$, and let Q_L be the set of states associated by π to the cuts in C_L . Notice that since $\mathbb{R}$ is complete, any subinterval of $\mathbb{R}$ is complete as well. Therefore, C_L also coincides with the set of cuts in $\widehat{(x, y)} \setminus \{y^-\}$ that do not admit a predecessor. Formally, Q_L is defined as follows:

$$Q_L = \{q \in S_\pi(x, y) \mid \exists z. x < z < y \text{ and } \pi(z^-) = q\}.$$

Consider a state $q \in Q_L \cup \{\pi(y^-)\}$. By contradiction, assume that either no left-limit transition of the form $P \rightarrow q$ is followed in π over the interval (x, y) , or that a left-limit transition of the form $P \rightarrow q$ with $P \neq S_\pi(x, y)$ is followed by π over (x, y) .

In the former case, if π does not follow any left-limit transition of the form $P \rightarrow q$ over (x, y) , then it can never reach q . Therefore, there exists a cut $z^- \in \widehat{(x, y)}$ that does not have a predecessor, such that $\lim_{z^-} \pi \rightarrow \pi(z^-)$ is not a transition in $\mathcal{A}$. This contradicts the fact that π is a valid path.

In the latter case, by definition of S_π , one has $P \subsetneq S_\pi(x, y)$. Let $s \in S_\pi(x, y)$ be such that $s \notin P$. Let $z \in (x, y]$ be such that $\pi(z^-) = q$ and $\lim_{(z^-)} \pi = P$. The definition of Q_L , and its non-emptiness, ensure that such a z exists. Let $z' \in (x, z)$ be such that for all z'' with $z' < z'' < z$, one has $\pi(z''^-) \neq s$ and $\pi(z''^+) \neq s$. The existence of such a z' is justified by the fact that the left-limit transition $P \rightarrow q$ must be followed in π to reach $\pi(z^-)$, otherwise s would belong to P . Now consider the interval (z', z) . This interval is non-empty, contained entirely within (x, y) , and such that $S_\pi(z', z) = P \subsetneq S$. This contradicts the original hypothesis about the π -stability of the interval (x, y) . $\square$

We are now ready to show that for any π -stable interval $(x, y) \subseteq \mathbb{R}$, the set of states $S_\pi(x, y)$ can always be rearranged into a structure satisfying the simple shuffle rule.

Lemma 5.4.5. *For any π -stable interval $(x, y) \subseteq \mathbb{R}$, the automaton $\mathcal{A}_j$ admits a corresponding transition $(\pi(x^+), S_\pi(x, y) \cup \{\pi(x^+), \pi(y^-)\}, \pi(y^-))$.*

Proof. Since the interval (x, y) is π -stable, by Lemma 5.4.4 we have:

- A starting and an ending state $p_I = \pi(x^+)$ and $p_F = \pi(y^-)$, such that
 - $\pi(x^+) \rightarrow S_\pi(x, y) \in \Delta^\ell$, and
 - $S_\pi(x, y) \rightarrow \pi(y^-) \in \Delta^\ell$.
- A set of states Q_L and a set of states Q_R , such that
 - $Q_L \cup Q_R = S_\pi(x, y)$,
 - for each $q \in Q_L$, there exists $q' \in Q_R$, $\alpha \in \Sigma$ such that $(q, \alpha, q') \in \Delta^s$,
 - for each $q \in Q_R$, there exists $q' \in Q_L$, $\alpha \in \Sigma$ such that $(q', \alpha, q) \in \Delta^s$,
 - for each $q \in Q_L$, the transition $S_\pi(x, y) \rightarrow q$ belongs to Δ^ℓ , and
 - for each $q \in Q_R$, the transition $q \rightarrow S_\pi(x, y)$ belongs to Δ^ℓ .

These are exactly the requirements for applying the simple shuffle rule, generating the transition $(p_I, S_\pi(x, y) \cup \{p_I, p_F\}, p_F)$. $\square$

Notice that unless $(-\infty, +\infty)$ is π -stable, no set of pairwise disjoint π -stable intervals can form a partition of $\mathbb{R}$, since such a partition would only be composed of open intervals. We denote by π -unstable point any real number that is not contained within a π -stable interval. We will show that any π -unstable point is necessarily an individual point, i.e., there does not exist a non-zero length interval of π -unstable points. Formally, let $\mathcal{S} = \bigcup_{(x,y) \text{ is } \pi\text{-stable}} (x, y)$, we have the following result.

Lemma 5.4.6. *For all $z \in \mathbb{R} \setminus \mathcal{S}$ and $\varepsilon > 0$, there exist $(x_1, y_1), (x_2, y_2) \subseteq \mathbb{R}$ such that:*

- $x_1 < y_1 < z < x_2 < y_2$,
- (x_1, y_1) and (x_2, y_2) are π -stable intervals,
- both $z - x_1 < \varepsilon$ and $y_2 - z < \varepsilon$.

Proof. Let $z \in \mathbb{R} \setminus \mathcal{S}$ and $\varepsilon > 0$. The intervals (x_1, y_1) and (x_2, y_2) can be constructed by initially setting $x_1 = z - \varepsilon/2$, $y_1 = z - \varepsilon/4$, $x_2 = z + \varepsilon/4$, and $y_2 = z + \varepsilon/2$. If (x_1, y_1) is not directly π -stable, then there must exist a subinterval (x'_1, y'_1) of (x_1, y_1) such that $S_\pi(x'_1, y'_1) \subsetneq S_\pi(x_1, y_1)$. If (x'_1, y'_1) is π -stable, then the construction stops. Otherwise the same operation is repeated after replacing (x_1, y_1) by (x'_1, y'_1) until reaching a π -stable subinterval. This will necessarily happen after a finite number of steps, since the set of states Q of $\mathcal{A}$ is finite. The same construction can be applied to find a suitable interval (x_2, y_2) . $\square$

The remainder of the proof of Theorem 5.4.1 boils down to showing that the π -stable intervals and the π -unstable points can be combined into a complete partition of $\mathbb{R}$, and that the different ways to combine them correspond to the rules implemented in our algorithm.

We now introduce the notion of π -pseudo-stable interval, corresponding to a non-empty open interval $(x, y) \subseteq \mathbb{R}$ such that $\mathcal{A}_j$ admits an open $\mathbb{R}$ -path from $\pi(x^+)$ to $\pi(y^-)$ labeled by $S_\pi(x, y) \cup \{\pi(x^+), \pi(y^-)\}$. We denote by $\bar{R}_{(x, y)}$ the set of all the open $\mathbb{R}$ -paths in $\mathcal{A}_j$ that start in $\pi(x^+)$ and end in $\pi(y^-)$, of length $n \leq |\Delta_j|^2$, labeled by $S_\pi(x, y) \cup \{\pi(x^+), \pi(y^-)\}$, where Δ_j is the set of transitions of the automaton $\mathcal{A}_j$. Bounding the length of the $\mathbb{R}$ -paths by $|\Delta_j|^2$ ensures that the set $\bar{R}_{(x, y)}$ is finite, while also guaranteeing that if $\mathcal{A}_j$ admits an open $\mathbb{R}$ -path from the state $\pi(x^+)$ to the state $\pi(y^-)$ labeled by $S_\pi(x, y) \cup \{\pi(x^+), \pi(y^-)\}$, then an analogous one exists in $\bar{R}_{(x, y)}$. The $|\Delta_j|^2$ bound provides an upper bound for the minimal length of an $\mathbb{R}$ -path visiting every transition in Δ_j .

Notice that any π -stable interval is also π -pseudo-stable, since by Lemma 5.4.5, an open $\mathbb{R}$ -path of length one (i.e., a single transition of $\mathcal{A}_j$) is associated with any π -stable interval. Furthermore, we call a π -pseudo-stable interval π -maximal if it is not contained within a greater π -pseudo-stable interval. Formally, an interval $(x, y) \subseteq \mathbb{R}$ is π -maximal if and only if

- $x = \min\{z \in \mathbb{R} \mid \text{there exists an open } \mathbb{R}\text{-path from } \pi(z^+) \text{ to } \pi(y^-) \text{ labeled by } S_\pi(z, y) \cup \{\pi(z^+), \pi(y^-)\}\}$, and
- $y = \max\{z \in \mathbb{R} \mid \text{there exists an open } \mathbb{R}\text{-path from } \pi(x^+) \text{ to } \pi(z^-) \text{ labeled by } S_\pi(x, z) \cup \{\pi(x^+), \pi(z^-)\}\}$.

Before moving to the case of the (reverse) infinite repetition rule, let us first introduce the following useful result.

Lemma 5.4.7. *Let $\ell \in [1, j-1]$, and let $\bar{r}$ be an open $\mathbb{R}$ -path in $\mathcal{A}_\ell$ from a state q to a state q' , labeled by a set of states P . For every $p \in P$, $\mathcal{A}_\ell$ also admits an open $\mathbb{R}$ -path $\bar{r}'$ from q to q' labeled by P of the form $(r_1, R_1, r'_1), \dots, (r_n, R_n, r'_n)$, such that:*

- $r_1 = q$ and $r'_n = q'$, and
- *there exists $i \in [1, n]$ such that either $r_i = p$ or $r'_i = p$.*

Proof. The existence of $\bar{r}$ implies that $\mathcal{A}$ admits a path μ from q to q' visiting exactly the states in P and labeled by a word w indexed by $\mathbb{R}$, such that $\mu(-\infty) = q$ and $\mu(+\infty) = q'$. For any $p \in P$, there exists by definition of μ a point $x \in \mathbb{R}$ such that either $\mu(x^-) = p$ or $\mu(x^+) = p$.

Consider the two subpaths of μ over the intervals $(-\infty, x)$ and $(x, +\infty)$. These paths are labeled by a word indexed by a non-empty open interval of $\mathbb{R}$, i.e., of order type λ . Therefore, there also exists a path μ' (resp. μ'') from q to $\mu(x^-)$ (resp. from $\mu(x^+)$ to q') labeled by a word w' (resp. w'') indexed by $\mathbb{R}$.

Given the inductive hypothesis on the completeness of $\mathcal{A}_\ell$, there must exist two open $\mathbb{R}$ -paths $\bar{r}_1$ and $\bar{r}_2$ in $\mathcal{A}_\ell$, such that $\bar{r}_1$ goes from q to $\mu(x^-)$ and $\bar{r}_2$ from $\mu(x^+)$ to q' . Furthermore, either $\bar{r}_1$ ends with a transition of the form (r_i, R_i, p) (if $\mu(x^-) = p$) or $\bar{r}_2$ starts with a transition of the form (p, R_i, r'_i) (if $\mu(x^+) = p$). Additionally, since μ is a valid path, there must also exist a transition

$(\mu(x^-), \alpha, \mu(x^+)) \in \Delta^s$ for some $\alpha \in \Sigma$. Therefore, one can concatenate $\bar{r}_1$ with $\bar{r}_2$ to obtain the open $\mathbb{R}$ -path of the desired form. The fact that the union of the labels of $\bar{r}_1$ and $\bar{r}_2$ is equal to P is immediate. $\square$

We are now ready to deal with the case where a set of π -pseudo-stable intervals can be combined thanks to the (reverse) infinite repetition rule.

Lemma 5.4.8. *Let (x, y) be a non-empty open interval in $\mathbb{R}$. If every interval (z, z') such that $x < z < z' < y$ is π -pseudo-stable, then (x, y) is π -pseudo-stable as well.*

Proof. Consider the subpath of π over the interval (x, y) . By definition, this subpath starts in $\pi(x^+)$ and ends in $\pi(y^-)$. It therefore ends with the left-limit transition $P \rightarrow \pi(y^-)$, where $P = \lim_{y^-} \pi$ and $|P| = k \leq j$. Let $z \in (x, y)$ be such that $S_\pi(z, y) = P$. Let $(p', \alpha, p) \in \Delta^s$ be such that, for every $y' < y$, π follows the transition (p', α, p) within the interval (y', y) , where $p, p' \in P$. Intuitively, this means that π follows this transition in every left neighborhood of y^- . Such a transition always exists since the set of successor transitions Δ^s is finite.

Let $(z_i)_{i \in \mathbb{N}}$ be a sequence of real numbers such that $\lim_{i \rightarrow +\infty} z_i = y$, and for every $i \in \mathbb{N}$, one has $z < z_i < z_{i+1} < y$, $\pi(z_i^-) = p'$, and $\pi(z_i^+) = p$. The existence of such a sequence is a direct consequence of the fact that π follows the transition (p', α, p) within every interval (y', y) with $y' < y$. By the infinite version of Ramsey's theorem [92], we can assume that this sequence satisfies $S_\pi(z_i, z_{i+1}) = P$, for all $i \in \mathbb{N}$.

We now show that there necessarily exists an open $\mathbb{R}$ -path from $\pi(z_0^+)$ to $\pi(y^-)$ labeled by $P \cup \{\pi(y^-)\}$ in $\mathcal{A}_k$, and therefore also in $\mathcal{A}_j$ since $k \leq j$. Notice that since $z_0 > z$, one necessarily has $P = S_\pi(z_0, y)$, with additionally $\pi(z_0^+) \in P$. Every interval (z_i, z_{i+1}) is π -pseudo-stable by hypothesis.

In the case where $\mathcal{A}_{k-1}$ admits an open $\mathbb{R}$ -path from p to p' labeled by P , the existence of such an open $\mathbb{R}$ -path in $\mathcal{A}_{k-1}$ together with the existence of a successor transition $(p', \alpha, p) \in \Delta^s$ implies that L_k necessarily contains the transition $(\pi(z_0^+), P \cup \{\pi(y^-)\}, \pi(y^-))$. Therefore, there exists an open $\mathbb{R}$ -path from $\pi(z_0^+)$ to $\pi(y^-)$ labeled by $P \cup \{\pi(y^-)\}$ in $\mathcal{A}_k$, i.e., (z_0, y) is π -pseudo-stable.

Consider now the case where $\mathcal{A}_k$ admits an open $\mathbb{R}$ -path $\bar{r}$ from p to p' labeled by P , but $\mathcal{A}_{k-1}$ does not. In this situation, a transition allowing to reach $\pi(y^-)$ from $p = \pi(z_0^+)$ may not be generated, since such a transition can only be created at Step k (since $|P| = k$), but relies on an open $\mathbb{R}$ -path that itself only appears at the same Step k . We now show that even if such transitions are not generated, there will nonetheless exist an open $\mathbb{R}$ -path from p to $\pi(y^-)$.

Let $(r, R, r') \in \Delta_k$ be a transition generated at Step k that is involved in $\bar{r}$, i.e., $\bar{r}$ is of the form $(r_1, R_1, r'_1), \dots, (r_n, R_n, r'_n)$ such that $r_1 = p$, $r'_n = p'$, and there exists $i \in [1, n]$ such that $(r_i, R_i, r'_i) = (r, R, r')$. In the eventuality where multiple transitions generated at Step k are involved in the open $\mathbb{R}$ -path $\bar{r}$ from p to p' , we assume that (r, R, r') is the leftmost one in $\bar{r}$.

We now show that for every rule that may have produced the transition (r, R, r') , some other transitions must also have been generated, resulting in the

existence of an open $\mathbb{R}$ -path in $\mathcal{A}_k$ from $\pi(z_0^+)$ to $\pi(y^-)$ labeled by $P \cup \{\pi(y^-)\}$. A first observation is that, independently from the rule that has produced the transition, R must be equal to P . Indeed, since (r, R, r') has been generated at step k , one has $|R| \geq k$. Furthermore, since this transition is involved in an open $\mathbb{R}$ -path labeled by P , one also has $R \subseteq P$. Since $|P| = k$, it follows that $P = R$.

Consider first the case where the transition (r, P, r') has been generated by an application of the (simple or Cantor) shuffle rule. Since the conditions of the shuffle rule were satisfied to generate the transition (r, P, r') , and since $\mathcal{A}$ admits the limit transition $P \rightarrow \pi(y^-)$, the construction of S_k also necessarily yielded the transition $(r, P \cup \{\pi(y^-)\}, \pi(y^-))$. By hypothesis, there exists an open $\mathbb{R}$ -path from p to p' labeled by P that involves the transition (r, P, r') , therefore an open $\mathbb{R}$ -path from p to $\pi(y^-)$ labeled by $P \cup \{\pi(y^-)\}$ can be easily constructed by taking the first part of $\bar{r}$ up to the transition (r, P, r') (excluded) and simply appending the transition $(r, P \cup \{\pi(y^-)\}, \pi(y^-))$.

Similarly, to deal with the case where the transition (r, P, r') has been generated by an application of the infinite repetition rule, one simply considers the conditions used to apply the infinite repetition rule to generate the transition (r, P, r') , this time replacing the endpoint state r' by $\pi(y^-)$, which is valid since $\mathcal{A}$ admits the limit transition $P \rightarrow \pi(y^-)$. It follows that L_k necessarily contains the transition $(r, P \cup \{\pi(y^-)\}, \pi(y^-))$, hence there exists an open $\mathbb{R}$ -path in $\mathcal{A}_k$ from $p = \pi(z_0^+)$ to $\pi(y^-)$ labeled by $P \cup \{\pi(y^-)\}$.

It remains to consider the case where the transition (r, P, r') has been generated by an application of the reverse infinite repetition rule. This case is more tricky as the application of the reverse infinite repetition rule does not straightforwardly implies that an analogous rule has been applied to reach $\pi(y^-)$.

Since the transition (r, P, r') has been generated at Step k by the reverse infinite repetition rule, the following conditions are necessarily satisfied:

- The automaton $\mathcal{A}_{k-1}$ admits an open $\mathbb{R}$ -path $\bar{r}'$ from a state r'' to the state r' labeled by P , such that $(r', \alpha, r'') \in \Delta^s$ for some $\alpha \in \Sigma$.
- The states r, r' and r'' belong to P , by definition of the label of $\mathbb{R}$ -paths for r' and r'' . If r did not belong to P , then the generation of the transition (r, P, r') would have relied on a limit transition involving the set $P \setminus \{r\}$ of cardinality $k - 1$, which could therefore not have been generated at Step k .

Since r belongs to P , by Lemma 5.4.7, $\mathcal{A}_{k-1}$ admits an open $\mathbb{R}$ -path from r'' to r' of the form $(q_1, Q_1, q'_1), \dots, (q_m, Q_m, q'_m)$ such that $q_1 = r''$, $q'_m = r'$, and such that either $q_\ell = r$ or $q'_\ell = r$, for some $\ell \in [1, m]$.

Consider the first case, i.e., assume that there exists $\ell \in [1, m]$ such that $q_\ell = r$. Given the existence of

- the open $\mathbb{R}$ -path $\bar{r}'$ in $\mathcal{A}_{k-1}$ labeled by P that starts in r'' and ends in r' ,
- a successor transition $(r', \alpha, r'') \in \Delta^s$, and

- the limit transition $P \rightarrow \pi(y^-)$,

it follows that at Step k , the construction of the set L_k must produce the transition $(q_j, P \cup \{\pi(y^-)\}, \pi(y^-))$ for every $j \in [1, m]$, by application of the infinite repetition rule, in particular for $j = \ell$. Notice that the way we defined the infinite repetition rule implies that its application only generates a transition for the state associated with the starting point of the first transition of the open $\mathbb{R}$ -path. But since this $\mathbb{R}$ -path actually forms a cycle, there exists a similar open $\mathbb{R}$ -path sharing the same label rooted at every state q_j visited by the cycle, on which the infinite repetition rule can also be applied.

Recall that the transition (r, P, r') appears in the $\mathbb{R}$ -path $\bar{r}$. By concatenating the first transitions composing $\bar{r}$ up to the $(i-1)$ -th one (i.e., stopping just before the occurrence of (r, P, r') , ensuring we end up in a state s such that there exists a transition $(s, \alpha, r) \in \Delta^s$ for some $\alpha \in \Sigma$) with the transition $(r, P, \pi(y^-))$, we obtain an open $\mathbb{R}$ -path of the desired form.

Now consider the second case, i.e., assume that there exists $\ell \in [1, m]$ such that $q'_\ell = r$. Considering the existence of

1. the open $\mathbb{R}$ -path $\bar{r}'$ in $\mathcal{A}_{k-1}$ labeled by P that starts in r'' and ends in r' ,
2. a successor transition $(r', \alpha, r'') \in \Delta^s$,
3. the limit transition $r \rightarrow P \in \Delta^\ell$ mandatory to apply the reverse infinite repetition rule and generate the transition (r, P, r') , and
4. the limit transition $P \rightarrow \pi(y^-)$,

it follows that:

- [(1),(2),(3)] At Step k , the construction of the set R_k must yield the transition (r, P, q'_j) for every $j \in [1, m]$, by application of the reverse infinite repetition rule, and in particular for $j = \ell$.
- [(1),(2),(4)] At Step k , the construction of the set L_k must yield the transition $(q_j, P \cup \{\pi(y^-)\}, \pi(y^-))$ for every $j \in [1, m]$, by application of the infinite repetition rule, and in particular for $j = 1$.

One can therefore build an open $\mathbb{R}$ -path that coincides with $\bar{r}$ (and therefore starts in p) up to the transition (r, P, r') , then follows the transition (r, P, r) before following the subpath of $\bar{r}'$ that starts right after the transition $(q_\ell, Q_\ell, q'_\ell)$ (which is possible because $q'_\ell = r$) and ends in r' , and finally follows the transition $(r'', P \cup \{\pi(y^-)\}, \pi(y^-))$.

A similar approach can be used to show that there exists a sequence $(z'_i)_{i \in \mathbb{N}}$ converging to x , such that there exists an open $\mathbb{R}$ -path from $\pi(x^+)$ to $\pi(z'_0^-)$ labeled by $S_\pi(x, z'_0) \cup \{\pi(x^+), \pi(z'_0^-)\}$ in $\mathcal{A}_j$. We therefore have two π -pseudo-stable intervals (x, z'_0) and (z_0, y) such that $x < z'_0 < z_0 < y$.

By hypothesis, the interval (z'_0, z_0) is also π -pseudo-stable. Therefore, by concatenating some open $\mathbb{R}$ -paths corresponding to the intervals (x, z'_0) , (z'_0, z_0) , and (z_0, y) in that order, we obtain an open $\mathbb{R}$ -path for the whole interval (x, y) , which entails that (x, y) is also π -pseudo-stable. $\square$

It remains to deal with π -unstable points that are neither a junction point between two consecutive π -pseudo-stable intervals, nor the limit of an infinite sequence of π -pseudo-stable intervals. A first step in that direction is to characterize the distribution of the π -maximal intervals.

Lemma 5.4.9. *The linear ordering between the π -maximal intervals of $\mathbb{R}$ is dense, i.e., for all π -maximal intervals $(x_1, y_1), (x_2, y_2)$ such that $y_1 < x_2$, there exists a π -maximal interval (x_3, y_3) such that $y_1 < x_3 < y_3 < x_2$.*

Proof. By contradiction, we prove that two such π -maximal intervals cannot be consecutive. Assume that it is the case, i.e., there exist two π -maximal intervals $(x_1, y_1), (x_2, y_2)$ such that $y_1 < x_2$, and such that there does not exist another maximal interval between y_1 and x_2 .

Since $y_1 < x_2$, the interval (y_1, x_2) is not empty, and any non-empty open interval of $\mathbb{R}$ contains a π -stable interval. Otherwise, it would mean that (y_1, x_2) only contains π -unstable points, which contradicts Lemma 5.4.6.

To any π -stable interval corresponds a greater π -maximal interval (it can be build by applying the rules generating π -pseudo-stable intervals, and then extending it as much as possible). However, the corresponding maximal interval cannot be contained strictly within (y_1, x_2) by hypothesis, and it cannot span before y_1 or beyond x_2 since $(x_1, y_1), (x_2, y_2)$ are both π -maximal by hypothesis, hence the contradiction. $\square$

Finally, the last step in proving Theorem 5.4.1 consists in showing that any non-trivial dense distribution of π -maximal intervals — i.e., containing at least two distinct π -maximal intervals, and therefore an infinity of them thanks to the density property — would be captured by the Cantor shuffle rule. This leads to the following result that states that there only exists a single π -maximal interval.

Lemma 5.4.10. *The only π -maximal interval of $\mathbb{R}$ is $\mathbb{R}$ itself.*

Proof. By contradiction, assume that there exist two distinct π -maximal intervals of $\mathbb{R}$. Let us pick two distinct π -maximal intervals (x_1, y_1) and (x_2, y_2) with $x_1 < y_1 < x_2 < y_2$, such that for any two π -maximal intervals $(a_1, b_1), (a_2, b_2)$ with $y_1 < a_1 < b_1 < a_2 < b_2 < x_2$, one has $S_\pi(y_1, x_2) = S_\pi(a_1, b_2)$. In other words, the set of states visited between the intervals (x_1, y_1) and (x_2, y_2) is minimal w.r.t. any subinterval of $\mathbb{R}$ containing at least two distinct π -maximal intervals. It is always possible to chose two such π -maximal intervals. Indeed, one can start from any two π -maximal intervals, and replace them if needed by two other intervals that appear between them, such that the set of states that π visits in the interval between them is strictly smaller. This operation can be repeated until reaching a fixed point, which will necessarily occur after a finite number of steps, since the set of states Q of $\mathcal{A}$ is finite.

Remember that any π -maximal interval is also π -pseudo-stable by definition. Therefore, for each π -maximal interval (a, b) , there exists an open $\mathbb{R}$ -path in $\mathcal{A}_j$, and more precisely in $\overline{R}_{(a,b)}$, from $\pi(a^+)$ to $\pi(b^-)$ labeled by $S_\pi(a, b) \cup \{\pi(a^+), \pi(b^-)\}$. By combining them with the successor transitions

$(\pi(a^-), \alpha_a, \pi(a^+))$ and $(\pi(b^-), \alpha_b, \pi(b^+))$ of Δ^s , these open $\mathbb{R}$ -paths can be turned into closed $\mathbb{R}$ -paths from $\pi(a^-)$ to $\pi(b^+)$, labeled by $S_\pi(a, b) \cup \{\pi(a^-), \pi(a^+), \pi(b^-), \pi(b^+)\}$. For each π -maximal interval (a, b) in (y_1, x_2) , let $\bar{r}^{[a, b]}$ be one such closed $\mathbb{R}$ -path. Using a similar argument as in the proof of Lemma 5.4.4, one can also show that both limit transitions $S_\pi(y_1, x_2) \rightarrow \pi(a^-)$ and $\pi(b^+) \rightarrow S_\pi(y_1, x_2)$ belong to Δ^ℓ . Furthermore, to each π -unstable point $z \in (y_1, x_2)$ we associate the successor transition $(\pi(z^-), \alpha_z, \pi(z^+)) \in \Delta^s$. Following the same approach as before, one can show that both limit transitions $S_\pi(y_1, x_2) \rightarrow \pi(z^-)$ and $\pi(z^+) \rightarrow S_\pi(y_1, x_2)$ belong to Δ^ℓ .

Notice that since Δ^s and each $\bar{R}_{(a, b)}$ are finite, both the set of successor transitions and the set of closed $\mathbb{R}$ -paths, respectively associated with the π -unstable points and π -maximal intervals in (y_1, x_2) , are finite as well.

Let P denote the set of states corresponding to the union of the labels of all closed $\mathbb{R}$ -paths $\bar{r}^{[a, b]}$ for every π -maximal interval (a, b) within (y_1, x_2) , together with the states involved in the successor transitions associated with all π -unstable points that lie between y_1 and x_2 . Formally, we have

$$P = \{q \mid z \text{ is a } \pi\text{-unstable point in } (y_1, x_2), \text{ and either } \pi(z^-) = q \text{ or } \pi(z^+) = q\} \\ \cup \{q \mid (a, b) \text{ is a } \pi\text{-maximal interval in } (y_1, x_2), R \text{ labels } \bar{r}^{[a, b]}, \text{ and } q \in R\}.$$

In order to show that the Cantor shuffle rule can be applied to the pair of transitions $\pi(y_1^+) \rightarrow S_\pi(y_1, x_2)$ and $S_\pi(y_1, x_2) \rightarrow \pi(x_2^-)$, it remains to show that $P = S_\pi(y_1, x_2)$ holds. It is clear that $P \subseteq S_\pi(y_1, x_2)$ holds. Indeed, for every π -unstable point z , we considered the states $\pi(z^-)$ and $\pi(z^+)$, which belong to $S_\pi(y_1, x_2)$ by definition. Similarly, the definition of the closed $\mathbb{R}$ -path $\bar{r}^{[a, b]}$ for each π -maximal interval (a, b) implies that its label is a subset of $S_\pi(y_1, x_2)$. In order to show $S_\pi(y_1, x_2) \subseteq P$, let us assume by contradiction that there exists $q \in S_\pi(y_1, x_2) \setminus P$. Then, there must exist $z \in (y_1, x_2)$ such that either $q = \pi(z^-)$ or $q = \pi(z^+)$, and z is neither a π -unstable point, nor contained within a π -maximal interval. Such a real number z cannot exist, as the set of π -unstable points combined with the set of π -maximal intervals (extended with their upper and lower bound to form closed intervals) form a partition of the interval (y_1, x_2) .

All the requirements of the Cantor shuffle rule are therefore met. It follows that the construction of the automaton $\mathcal{A}_j$ yields the transition $(\pi(y_1^+), S_\pi(y_1, x_2) \cup \{\pi(y_1^+), \pi(x_2^-)\}, \pi(x_2^-))$. Since (x_1, y_1) (resp. (x_2, y_2)) is π -pseudo-stable, there exists an open $\mathbb{R}$ -path $\bar{r}_1$ (resp. $\bar{r}_2$) from $\pi(x_1^+)$ to $\pi(y_1^-)$ (resp. from $\pi(x_2^+)$ to $\pi(y_2^-)$). By concatenating $\bar{r}_1$, the transition $(\pi(y_1^+), S_\pi(y_1, x_2) \cup \{\pi(y_1^+), \pi(x_2^-)\}, \pi(x_2^-))$ of $\mathcal{A}_j$, and $\bar{r}_2$ (in that order), we obtain an open $\mathbb{R}$ -path from $\pi(x_1^+)$ to $\pi(y_2^-)$ labeled by $S_\pi(x_1, y_2) \cup \{\pi(x_1^+), \pi(y_2^-)\}$. This implies that (x_1, y_2) is π -pseudo-stable, which contradicts the assumption that (x_1, y_1) and (x_2, y_2) are π -maximal intervals. $\square$

This concludes the proof of Theorem 5.4.1.

The proof of correctness developed in this section can be straightforwardly adapted to the algorithm of Section 5.3, dealing with the reachability problem

for words indexed by $\mathbb{Q}$.

5.5 Additional Remarks

Optimizations Since the publication of this paper, an efficient implementation of the reachability algorithms we present in this chapter has been proposed in [18]. The optimization the authors consider roughly consists in introducing alternate versions of the shuffle and (reverse) infinite repetition rules that, at each step $i \in [1, M]$, generate a smaller set of transitions in the automaton $\mathcal{A}_i$, sufficient to correctly detect the existence of an accepting path in the input automaton.

Erratum In the presentation of the state of the art (Section 5.1), we mentioned that (at the time of publication) we did not know of a proof of the sketched non-emptiness test over all linear orderings proposed by Cristau in [38]. Since then, we have found a proof in Cristau’s PhD thesis [39] (written in French). It however contains a small error which makes the algorithm oblivious of paths indexed by linear orderings that contain an infinite, dense, and incomplete subordering.

Generalizing this Approach It may be of interest to characterize classes of domains for which a dedicated non-emptiness test can be performed, beyond the domains $\mathbb{R}$ and $\mathbb{Q}$ investigated here. For instance, to decide a theory interpreted over domains with a cofinality (respectively coinitality) greater than or equal to ω_1 (respectively $-\omega_1$), a suitable non-emptiness test should account for the fact that automata on linear ordering can filter out words indexed by orderings with an uncountable cofinality or coinitality, as illustrated in the proof of Lemma 4.2.2.

This problem is strongly connected with establishing the precise expressive power of these automata on linear orderings, and their ties to logic, which have been investigated in [5].

Chapter 6

First-Order Universal Quantification

We start this chapter by formally introducing the notion of first-order universal quantification for automata recognizing models of $\text{MFO}(<)$ formulas. We then describe a procedure to directly compute this operation for the particular case of infinite-word automata. Finally, we discuss leads on how to design an analogous procedure for automata on linear orderings.

6.1 Principles

Definitions Given a word u indexed by a linear ordering defined over the alphabet $\{0, 1\}^{n-1+m}$ corresponding to the encoding of $n-1$ first-order variables and m second-order variables (with $n > 0$ and $m \geq 0$), a position $j \in |u|$, and $i \in [1, n]$, $u \oplus [x_i = j]$ denotes the word indexed by $|u|$ over the alphabet $\{0, 1\}^{n+m}$, satisfying $\Theta_{\neq i}(u \oplus [x_i = j]) = u$, and $\Theta_{=i}(u \oplus [x_i = j]) = 0^{J^-} \cdot 1 \cdot 0^{J^+}$, where $J^- = \{e \in |u| \mid e < j\}$, and $J^+ = \{e \in |u| \mid j < e\}$. Intuitively, the word $u \oplus [x_i = j]$ can be viewed as the word u in which we insert a fresh first-order tape at index i whose content encodes the element $j \in |u|$.

As an illustration, consider the word $u = ((0, 1) \cdot (1, 0))^\omega$ that corresponds to the encodings of the pair $(N_{\text{odd}}, N_{\text{even}})$ of subsets of $\mathbb{N}$, denoting respectively the set of odd and even numbers. The word $v = u \oplus [x_1 = 29]$ is defined as $v = ((0, 0, 1) \cdot (0, 1, 0))^{14} \cdot (0, 0, 1) \cdot (1, 1, 0) \cdot ((0, 0, 1) \cdot (0, 1, 0))^\omega$.

We are now ready to define the effect of first-order universal quantification on (n, m) -languages. For a (n, m) -language L and $i \in [1, n]$, the operator $\forall x_i$ applied to L yields the $(n-1, m)$ -language $\forall x_i(L)$ defined as

$$\forall x_i(L) = \{u \in (\{0, 1\}^{n-1+m})^\diamond \mid \forall j \in |u|, u \oplus [x_i = j] \in L\}.$$

Analogously to first-order universal quantification, one can also define the first-order existential quantification of languages. Given an (n, m) -language L

and $i \in [1, n]$, we define the operator $\exists x_i$ which when applied to L , yields the $(n-1, m)$ -language $\exists x_i(L)$ defined as

$$\exists x_i(L) = \{u \in (\{0, 1\}^{n-1+m})^\diamond \mid \exists j \in |u|, u \oplus [x_i = j] \in L\}.$$

We already mentioned in Chapter 4 that the operation on automata corresponding to existential quantification consists in erasing the tape associated with x_i altogether from the input (n, m) -automaton accepting the language L .

Let $U_{(n,m)}$ be the *universal* (n, m) -language, i.e., the language comprising all words corresponding to valid encodings of the values of n first-order variables and m second-order variables. More precisely, one has

$$U_{(n,m)} = \{u \in (\{0, 1\}^{n+m})^\diamond \mid \forall i \in [1, n]. \Theta_{=i}(u) \in 0^\diamond \cdot 1 \cdot 0^\diamond\}.$$

Consider the operator on (n, m) -languages $\neg$ defined by $\neg L = U_{(n,m)} \setminus L$. We can notice that for every (n, m) -language L , the language $\forall x_i(L)$ coincides with $\neg(\exists x_i(\neg L))$, which furthermore highlights the correspondence between the operators introduced in this section, and their corresponding logical counterparts.

Closure Since there exists a construction to compute an automaton accepting the language $\exists x_i(L(\mathcal{A}))$ from any (n, m) -automaton $\mathcal{A}$, it follows that the class of all (n, m) -languages of words indexed by arbitrary linear orderings, for any $n, m \geq 0$, is closed under first-order existential quantification. On the other hand, it is an open question whether the same can be said for first-order universal quantification.

Conjecture 1. *Over the class of all linear orderings, the class of all (n, m) -languages is closed under first-order universal quantification.*

Remark. Notice that the conjecture holds over the domain $\mathbb{N}$, and over the class of countable and scattered linear orderings. Indeed, it follows from the equivalence $\forall x_i. \varphi \equiv \neg \exists x_i. \neg \varphi$ and the fact that complementation preserves regularity over these domains.

6.2 Quantification Over Natural Numbers

In this section, we present a method to directly compute the effect of first-order universal quantification on an infinite-word automaton recognizing the set of models of a formula. More precisely, we consider Büchi automata, and the encoding of models using unary notation described in Chapter 4.

The remainder of this section reproduces parts — i.e., Sections 4.1 to 4.7 — of the following article [14]:

Bernard Boigelot, Pascal Fontaine, and Baptiste Vergain. Universal first-order quantification over automata. *International Journal of Foundations of Computer Science*, 37(01):5–21, 2026.

Some minor changes in notations have been introduced to match the ones provided of this work, and additional notations, introduced in the preliminaries of [14], are introduced below.

For a finite-word or infinite-word automata $\mathcal{A}$, we denote by $L(\mathcal{A}, q)$ the language defined as the set of all words accepted from a state q of $\mathcal{A}$. An automaton accepting $L(\mathcal{A}, q)$ can be derived from $\mathcal{A}$ by setting the set of initial states I equal to $\{q\}$. The language of finite words w read by paths from q_1 to q_2 , with q_1 and q_2 two states of $\mathcal{A}$, is denoted by $L(\mathcal{A}, q_1, q_2)$; a finite-word automaton accepting this language can be obtained from $\mathcal{A}$ by setting I equal to $\{q_1\}$ and F equal to $\{q_2\}$.

Furthermore, in the original paper, we introduce the following running example to illustrate our procedure.

Example 10. Consider the MFO($\mathbb{N}, <$) formula $\varphi(x_1, x_2, X_1) = x_1 < x_2 \wedge X_1(x_2)$. An example of automata recognizing the set of models of the formulas φ and $\exists x_2. \varphi$ are given in, respectively, Figure 6.1a and Figure 6.1b. We will later illustrate our procedure by constructing an automaton recognizing the set of models of the formula $\forall x_1. \exists x_2. \varphi$. Intuitively, the formula $\forall x_1. \exists x_2. \varphi$ expresses that the set associated with the second-order variable X_1 is unbounded.

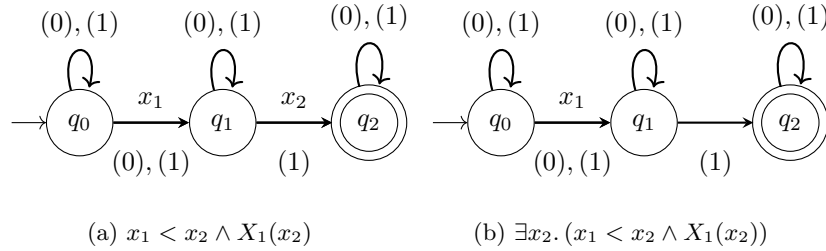


Figure 6.1: Automata recognizing sets of models.

6.2.1 Principles

Let $\varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$, with $k > 0$ and $\ell \geq 0$, be a formula. Given an automaton $\mathcal{A}$ recognizing the set of models of the formula φ and $i \in [1, k]$, our goal is to compute an automaton $\mathcal{A}'$ accepting $\forall x_i(L(\mathcal{A}))$. Since we are in a simpler setting where the only domain considered is $\mathbb{N}$, the language $\forall x_i(L(\mathcal{A}))$ can be equivalently redefined as

$$\forall x_i(L(\mathcal{A})) = \{u \in \{0, 1\}^{+\omega} \mid \forall n \in \mathbb{N}, u \oplus [x_i = n] \in L(\mathcal{A})\}.$$

Conceptually, we can also define for each $n \in \mathbb{N}$ the language $S_n = \{u \mid u \oplus [x_i = n] \in L\}$, which yields $\forall x_i(L(\mathcal{A})) = \bigcap_{n \in \mathbb{N}} S_n$.

An automaton $\mathcal{A}'$ accepting $\forall x_i(L(\mathcal{A}))$ can be obtained as follows. Each language S_n , with $n \in \mathbb{N}$, is accepted by an automaton $\mathcal{A}_n$ derived from $\mathcal{A}$ by

restricting the transitions associated to x_i to be followed only after having read exactly n symbols. In other words, the accepting runs of $\mathcal{A}_n$ correspond to the accepting runs of $\mathcal{A}$ that satisfy this condition. After imposing this restriction, the variable x_i is removed from the set of variables managed by the automaton, i.e., the operator $\Theta_{\neq i}$ is applied to the language that this automaton accepts, so as to get $S_n = L(\mathcal{A}_n)$. The automaton $\mathcal{A}'$ then corresponds to the infinite intersection product of the automata $\mathcal{A}_n$ for all $n \in \mathbb{N}$, i.e., an automaton that accepts the infinite intersection $\bigcap_{n \in \mathbb{N}} S_n$. We show in the next section how to build $\mathcal{A}'$ by means of a finite computation.

6.2.2 Construction

The idea of the construction is to make $\mathcal{A}'$ simulate the join behavior of the automata $\mathcal{A}_n$, for all $n \in \mathbb{N}$, on the same input words. This can be done by making each state of $\mathcal{A}'$ correspond to one state q_n in each $\mathcal{A}_n$, i.e., to an infinite tuple $(q_0, q_1, \dots)$. By definition of $\mathcal{A}_n$, there exists a mapping $\mu : Q_n \rightarrow Q$, where Q_n and Q are respectively the sets of states of $\mathcal{A}_n$ and $\mathcal{A}$, such that whenever a run of $\mathcal{A}_n$ visits q_n , the corresponding run of $\mathcal{A}$ on the same input word visits $\mu(q_n)$.

If two automata $\mathcal{A}_{n_1}$ and $\mathcal{A}_{n_2}$, with $n_1, n_2 \in \mathbb{N}$, are (respectively) in states q_{n_1} and q_{n_2} such that $\mu(q_{n_1}) = \mu(q_{n_2})$, then they share the same future behaviors, except for the requirement to follow a transition associated to x_i after having read (respectively) n_1 and n_2 symbols. It follows that the states of $\mathcal{A}'$ can be characterized by sets of states of $\mathcal{A}$: the infinite tuple $(q_0, q_1, \dots)$ is described by the set $\{\mu(q_i) \mid i \in \mathbb{N}\}$. Each element of this set represents the current state of one or several automata among the $\mathcal{A}_n$. This means that the number of these automata that are in this current state is not counted. We will establish that this abstraction is precise and leads to a correct construction.

During a run of $\mathcal{A}'$, each transition with a label other than ε must correspond to a transition reading the same symbol in every automaton $\mathcal{A}_n$, which in turn can be mapped to a transition of $\mathcal{A}$. In the automaton $\mathcal{A}_n$ for which n is equal to the number of symbols already read during the run, this transition of $\mathcal{A}$ is necessarily associated to x_i , by definition of $\mathcal{A}_n$. It follows that every transition of $\mathcal{A}'$ with a non-empty label is characterized by a set of transitions of $\mathcal{A}$, among which one of them is associated to x_i .

We are now ready to describe formally the construction of $\mathcal{A}'$, leaving for the next section the problem of determining which of its runs should be accepting or not: from the automaton $\mathcal{A} = (\Sigma, Q, \Delta, Q_0, F)$, we construct $\mathcal{A}' = (\Sigma', Q', \Delta', Q'_0, F')$ such that

- $\Sigma' = \Theta_{\neq i}(\Sigma)$.
- $Q' = 2^Q \setminus \{\emptyset\}$.
- Δ' contains
 - the transitions $(\bar{q}_1, (a'_1, \dots, a'_{k+\ell-1}), \bar{q}_2)$ for which there exists a set $T \subseteq \Delta$ that satisfies the following conditions:

- * $\bar{q}_1 = \{q_1 \mid (q_1, (a_1, \dots, a_{k+\ell}), q_2) \in T\}$.
- * $\bar{q}_2 = \{q_2 \mid (q_1, (a_1, \dots, a_{k+\ell}), q_2) \in T\}$.
- * For all $(q_1, (a_1, \dots, a_{k+\ell}), q_2) \in T$, one has $a'_j = a_j$ for all $j \in [1, i-1]$, and $a'_j = a_{j+1}$ for all $j \in [i, k+\ell-1]$.
- * There exists exactly one $(q_1, (a_1, \dots, a_{k+\ell}), q_2) \in T$ such that $a_i = 1$.
- the transitions $(\bar{q}_1, \varepsilon, \bar{q}_2)$ for which there exists a transition $(q_1, \varepsilon, q_2) \in \Delta$ such that
 - * $q_1 \in \bar{q}_1$.
 - * $\bar{q}_2 = \bar{q}_1 \cup \{q_2\}$ or $\bar{q}_2 = (\bar{q}_1 \setminus \{q_1\}) \cup \{q_2\}$.
- $Q'_0 = 2^{Q_0} \setminus \{\emptyset\}$.
- $F' = Q'$ for now. The problem of characterizing more finely the accepting runs will be addressed in the next section.

The rule for the transitions $(\bar{q}_1, (a'_1, \dots, a'_{k+\ell-1}), \bar{q}_2)$ ensures that for each $q_1 \in \bar{q}_1$, each automaton $\mathcal{A}_n$ that is simulated by $\mathcal{A}'$ has the choice of following any possible transition originating from q_1 that has a label consistent with $(a'_1, \dots, a'_{k+\ell-1})$. One such automaton must nevertheless follow a transition associated to the quantified variable x_i . The rule for the transitions $(\bar{q}_1, \varepsilon, \bar{q}_2)$ expresses that one automaton $\mathcal{A}_n$, or any number of identical copies of this automaton, must follow a transition labeled by ε , while the other automata stay in their current state.

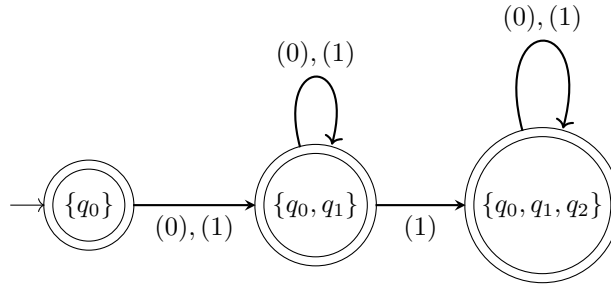


Figure 6.2: First step of construction for $\forall x_1 \exists x_2 (x_1 < x_2 \wedge X_1(x_2))$.

As an example, applying this construction to the automaton in Figure 6.1b, as a first step of the computation of a representation of $\forall x_1 \exists x_2 (x_1 < x_2 \wedge X_1(x_2))$, yields the automaton given in Figure 6.2. For the sake of clarity, unreachable states and states from which the accepted language is empty are not depicted.

6.2.3 A Criterion for Accepting Runs

The automaton $\mathcal{A}'$ defined in the previous section simulates an infinite combination of automata $\mathcal{A}_n$, for all $n \in \mathbb{N}$. By construction, every accepting run of

this infinite combination corresponds to a run of $\mathcal{A}'$.

The reciprocal property is not true, in the sense that there may exist a run of $\mathcal{A}'$ that does not match an accepting run of the infinite combination of automata $\mathcal{A}_n$. Consider for instance a run of the automaton in Figure 6.2 that ends up cycling in the state $\{q_0, q_1, q_2\}$, reading 0^ω from that state. Recall that for this example, the automaton $\mathcal{A}$ that undergoes the universal quantification operation is the one given in Figure 6.1b. The run that we have considered can be followed in $\mathcal{A}'$, but cannot be accepting in every $\mathcal{A}_n$. Indeed, in this example, the transition of $\mathcal{A}_n$ reading the $(n+1)$ -th symbol of the run corresponds, by definition of this automaton, to the transition of $\mathcal{A}$ that is associated to the quantified variable x_1 . By the structure of $\mathcal{A}$, this transition is necessarily followed later in any accepting run by one that reads the symbol 1, which implies that no word of the form $u \cdot 0^\omega$, with $u \in \{0, 1\}^*$, can be accepted by a run of $\mathcal{A}_n$ such that $n \geq |u|$. This represents the fact that the words accepted by all $\mathcal{A}_n$ correspond to the encodings of predicates that are true infinitely often.

One thus needs a criterion for characterizing the runs of $\mathcal{A}'$ that correspond to combinations of accepting runs in all automata $\mathcal{A}_n$.

It is known [78, 26] that two ω -regular languages over the alphabet Σ are equal iff they share the same set of *ultimately periodic* words, i.e., words of the form $u \cdot v^\omega$ with $u \in \Sigma^*$ and $v \in \Sigma^+$. It follows that it is sufficient to characterize the accepting runs of $\mathcal{A}'$ that read ultimately periodic words. The automaton $\mathcal{A}'$ accepts a word $u \cdot v^\omega$ iff every $\mathcal{A}_n$, with $n \in \mathbb{N}$, admits an accepting run that reads this word. Note that such a run also matches a run of $\mathcal{A}$, and that this run of $\mathcal{A}$ always ends up following a cycle from an accepting state to itself.

Our solution takes the following form. For each state q of $\mathcal{A}$, we define a language $U_q \subseteq \Sigma^+$ of non-empty words u such that $\mathcal{A}$ accepts u^ω from q , after dismissing the input tape associated to the quantified variable x_i . The alphabet Σ is thus equal to $\{0, 1\}^{k+\ell-1}$. Remember that each state $\bar{q}$ of $\mathcal{A}'$ is defined as a subset of states of $\mathcal{A}$, corresponding to the current states in the combination of copies of $\mathcal{A}$ that are jointly simulated by $\mathcal{A}'$. In order for the word u^ω to be accepted by $\mathcal{A}'$ from $\bar{q}$, it should therefore be accepted by $\mathcal{A}$ from each state $q \in \bar{q}$, i.e., u must belong to all the languages U_q such that $q \in \bar{q}$.

It must also be possible to read u^ω from the state $\bar{q}$ of $\mathcal{A}'$. We impose a stronger condition, by requiring that there exists a cycle from $\bar{q}$ to itself labeled by u . We will establish that this condition leads to a correct acceptance criterion.

In summary, the language $U_{\bar{q}}' = L(\mathcal{A}', \bar{q}, \bar{q}) \cap \bigcap_{q \in \bar{q}} U_q$ characterizes the words u such that u^ω must be accepted from the state $\bar{q}$ of $\mathcal{A}'$. Note that for this property to hold, it is not necessary for the language U_q to contain all words u such that $u^\omega \in \Theta_{\neq i}(L(\mathcal{A}, q))$, but only some number of copies u^p , where $p > 0$ is bounded, of each such u . In other words, the finite words u whose infinite repetition is accepted from q do not have to be the shortest possible ones.

Once the language $U_{\bar{q}}'$ has been obtained, we build a *widget*, in the form of an infinite-word automaton accepting $(U_{\bar{q}}')^\omega$, along the state $\bar{q}$ of $\mathcal{A}'$, and add a transition labeled by ε from $\bar{q}$ to the initial state of this widget. This ensures that every path that ends up in $\bar{q}$ can be suitably extended into an accepting run. Such a widget does not have to be constructed for every state $\bar{q}$ of $\mathcal{A}'$:

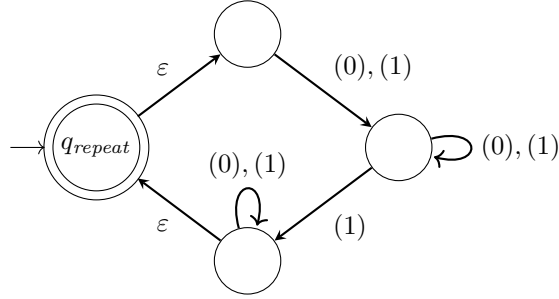
since the goal is to accept from $\bar{q}$ words of the form u^ω , we can require that at least one state $q \in \bar{q}$ is accepting in $\mathcal{A}$. We then only build widgets for the states $\bar{q}$ that satisfy this requirement.

6.2.4 Computation Steps

The procedure for modifying $\mathcal{A}'$ in order to make it accept the runs that match those of the infinite combination of automata $\mathcal{A}_n$, outlined in the previous section, can be carried out by representing the regular languages U_q and U'_q by finite-state automata. The construction proceeds as follows:

1. For each state $q \in Q$ of $\mathcal{A}$, build a finite-word automaton $\mathcal{A}_q$ that accepts all the non-empty words u for which there exists a path $q \xrightarrow{v} q$ of $\mathcal{A}$ that visits at least one accepting state $q_F \in F$, such that $u = \Theta_{\neq i}(v)$. This automaton can be constructed in a similar way as one accepting $\Theta_{\neq i}(L(\mathcal{A}, q, q))$, keeping one additional bit of information in its states for determining whether an accepting state has already been visited or not.
2. For each pair of states $q_1, q_2 \in Q$ of $\mathcal{A}$, build a finite-word automaton $\mathcal{A}_{q_1, q_2}$ accepting the language $\Theta_{\neq i}(L(\mathcal{A}, q_1, q_2))$.
3. For each state $q \in Q$ of $\mathcal{A}$, build an automaton $\mathcal{A}_{U_q} = \bigcup_{r \in Q} (\mathcal{A}_{q, r} \cap \mathcal{A}_r)$ accepting the finite-word language U_q .
4. For each state $\bar{q}$ of $\mathcal{A}'$ such that $\bar{q} \cap F \neq \emptyset$, where F is the set of accepting states of $\mathcal{A}$, build a finite-word automaton $\mathcal{A}'_{U'_q} = \mathcal{A}'_{\bar{q}} \cap \bigcap_{q \in \bar{q}} \mathcal{A}_{U_q}$ accepting U'_q , where $\mathcal{A}'_{\bar{q}}$ is an automaton accepting $L(\mathcal{A}', \bar{q}, \bar{q})$.
5. Then, turn each automaton $\mathcal{A}'_{U'_q}$ into an infinite-word automaton $\mathcal{A}'_{(U'_q)^\omega}$ accepting $(U'_q)^\omega$:
 - (a) Create a new state q_{repeat} .
 - (b) Add a transition $(q_{repeat}, \varepsilon, q_0)$ for each initial state q_0 , and a transition $(q_F, \varepsilon, q_{repeat})$ for each accepting state q_F , of $\mathcal{A}'_{U'_q}$.
 - (c) Make q_{repeat} the only initial and accepting state of $\mathcal{A}'_{(U'_q)^\omega}$.
6. For each state $\bar{q}$ of $\mathcal{A}'$ considered at Step 4, add the widget $\mathcal{A}'_{U'_q}$ alongside $\bar{q}$, by incorporating its sets of states and transitions into those of $\mathcal{A}'$, and adding a transition $(\bar{q}, \varepsilon, q_{repeat})$. In the resulting automaton, mark as the only accepting states the states q_{repeat} of all widgets.

We call $\mathcal{A}''$ the automaton constructed by this procedure. This automaton accepts the language $\forall x_i(L(\mathcal{A}))$.

Figure 6.3: Widget for the state $\{q_0, q_1, q_2\}$.

6.2.5 Illustration

We illustrate the construction described in Section 6.2.4 on the automaton $\mathcal{A}'$ in Figure 6.2, obtained after the first step of the universal quantification procedure applied to the automaton $\mathcal{A}$ in Figure 6.1b. We assume that the reader is familiar with the notation of regular and ω -regular languages by regular expressions.

We obtain automata accepting the following languages at the successive steps of the procedure:

1. $L(\mathcal{A}_{q_0}) = L(\mathcal{A}_{q_1}) = \emptyset$ and $L(\mathcal{A}_{q_2}) = (0 + 1)^+$.
2. $L(\mathcal{A}_{q_0, q_0}) = L(\mathcal{A}_{q_1, q_1}) = L(\mathcal{A}_{q_2, q_2}) = (0 + 1)^*$, $L(\mathcal{A}_{q_0, q_1}) = (0 + 1)^+$, $L(\mathcal{A}_{q_0, q_2}) = (0 + 1)^+ 1 (0 + 1)^*$, $L(\mathcal{A}_{q_1, q_2}) = (0 + 1)^* 1 (0 + 1)^*$, and $L(\mathcal{A}_{q_j q_{j'}}) = \emptyset$ for all other pairs $(q_j, q_{j'})$ of states.
3. $L(\mathcal{A}_{U_{q_0}}) = (0 + 1)^+ 1 (0 + 1)^*$, $L(\mathcal{A}_{U_{q_1}}) = (0 + 1)^* 1 (0 + 1)^*$, and $L(\mathcal{A}_{U_{q_2}}) = (0 + 1)^+$.
4. $L(\mathcal{A}'_{U'_{\{q_1, q_2\}}}) = \emptyset$ and $L(\mathcal{A}'_{U'_{\{q_0, q_1, q_2\}}}) = (0 + 1)^+ 1 (0 + 1)^*$.
5. The widget $\mathcal{A}'_{U'_{\{q_0, q_1, q_2\}}}$ is given in Figure 6.3.
6. The resulting automaton $\mathcal{A}''$ is shown in Figure 6.4. For the sake of clarity, the states from which the accepted language is empty have been removed.

6.2.6 Proof of Correctness

Let us show that the automaton $\mathcal{A}''$ constructed according to the procedure developed in Section 6.2.4 simulates exactly the infinite intersection of the automata $\mathcal{A}_n$, for all $n \geq 0$, as defined in Section 6.2.2. We first establish that every word accepted by $\mathcal{A}''$ is also accepted by $\mathcal{A}_n$ for each $n \geq 0$, after dismissing the input tape associated to the quantified variable.

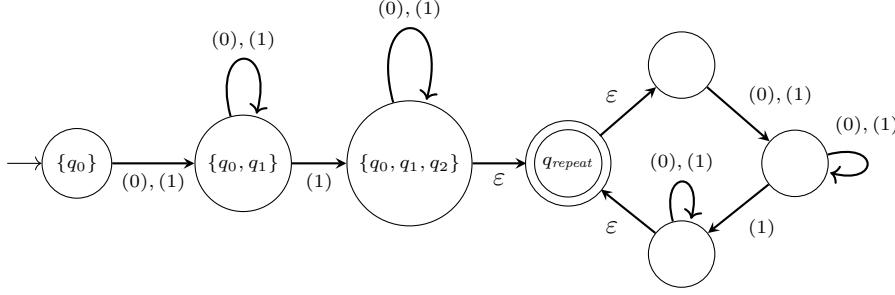


Figure 6.4: Automaton recognizing the set of models of $\forall x_1 \exists x_2 (x_1 < x_2 \wedge X_1(x_2))$.

Note that, since we know that applying a first-order universal quantifier to an automaton recognizing a set of models preserves ω -regularity (cf. the remark in Section 6.1), and two ω -regular languages are equal iff they share the same ultimately periodic words [78, 26], it is sufficient to consider ultimately periodic words. We will rely on the following lemma.

Lemma 6.2.1. *If an infinite-word automaton admits a path that reads an ultimately periodic word $w = u \cdot v^\omega$, with $|v| > 0$, originating from a state q_0 and visiting infinitely many times a state q , then it admits a path of the form $q_0 \xrightarrow{u_1} q \xrightarrow{u_2} q \xrightarrow{u_3} q \xrightarrow{u_4} \dots$, with $|u_2| > 0$, that reads w .*

Proof. The automaton admits a path

$$q_0 \xrightarrow{u'_1} q \xrightarrow{u'_2} q \xrightarrow{u'_3} q \xrightarrow{u'_4} \dots$$

that reads w . For every $i \in \mathbb{N}_{\geq 2}$, we define $v_i = u'_2 u'_3 \dots u'_i$. Modulo $|v|$, the range of possible values for $|v_i|$ is finite, hence there exist i, j such that $i < j$ and $|v_i| = |v_j| \pmod{|v|}$. This implies $|u'_{i+1} u'_{i+2} \dots u'_j| = k|v|$ for some $k \in \mathbb{N}_{>0}$. We then obtain the result by setting $u_1 = u'_1 u'_2 \dots u'_i$ and $u_2 = u'_{i+1} u'_{i+2} \dots u'_j$. $\square$

We are now ready to establish our first result.

Theorem 6.2.1. *For every ultimately periodic word $w'' \in L(\mathcal{A}'')$ and $n \geq 0$, there exists an accepting run ρ of $\mathcal{A}$ such that*

- *the word w read by ρ satisfies $\Theta_{\neq i}(w) = w''$, and*
- *the $(n+1)$ -th transition with a non-empty label followed by ρ is associated to the quantified variable x_i . (In other words, such a run ρ also exists for the automaton $\mathcal{A}_n$.)*

Proof. The only accepting states of $\mathcal{A}''$ are the states q_{repeat} , and those are not reachable from each other. Thus, for every run of $\mathcal{A}''$ that accepts w'' , there exists a single state q_{repeat} that this run visits infinitely often. By Lemma 6.2.1, there exists a run ρ'' of $\mathcal{A}''$ accepting w'' that is of the form

$$\rho'' : \bar{q}_0 \xrightarrow{u_1} \bar{q} \xrightarrow{\varepsilon} q_{repeat} \xrightarrow{u_2} q_{repeat} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} \dots,$$

where $\bar{q}_0$ is an initial state of $\mathcal{A}''$, and $w'' = u_1 u_2 (u_3)^\omega$ with $|u_2| > 0$ and $|u_3| > 0$.

By construction of the widget associated to $\bar{q}$, if there exists a non-empty word u such that $\mathcal{A}''$ admits the path $q_{repeat} \xrightarrow{u} q_{repeat}$, then the path $\bar{q} \xrightarrow{u} \bar{q}$ exists as well in $\mathcal{A}'$, and therefore in $\mathcal{A}''$. (Securing this property was the motivation behind the restriction discussed in the definition of U_q^L in Section 6.2.3.) Therefore, $\mathcal{A}''$ admits the paths $\bar{q} \xrightarrow{u_3} \bar{q}$ and $\bar{q} \xrightarrow{u_3} \bar{q}$, and there exists a run ρ_1'' of $\mathcal{A}''$ accepting w that is of the form

$$\rho_1'' : \bar{q}_0 \xrightarrow{u_1} \bar{q} \xrightarrow{u_2} \bar{q} \xrightarrow{u_3} \bar{q} \xrightarrow{\varepsilon} q_{repeat} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} \dots$$

For every $p > 0$, the path $\bar{q} \xrightarrow{u_3} \bar{q}$ that appears in this run can be repeated p times, which results in a path

$$\rho_p'' : \bar{q}_0 \xrightarrow{u_1} \bar{q} \xrightarrow{u_2} \bar{q} \xrightarrow{(u_3)^p} \bar{q} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} \dots$$

that accepts w'' . We then choose p such that $|u_1 u_2 (u_3)^p| > n$, in order to ensure that the $(n+1)$ -th symbol read by ρ_p'' corresponds to a transition located in $\mathcal{A}'$ and not in any widget.

By construction of $\mathcal{A}'$ (cf. Section 6.2.2), there exists a path π of $\mathcal{A}$ of the form $\pi : q_0 \xrightarrow{u} q_j$, where q_0 is an initial state, $q_j \in \bar{q}$, $\Theta_{\neq i}(u) = u_1 \cdot u_2 \cdot (u_3)^p$, and the $(n+1)$ -th symbol read by π is associated to the quantified variable x_i . Furthermore, since the widget associated to $\bar{q}$ admits the path $q_{repeat} \xrightarrow{u_3} q_{repeat}$, we have $u_3 \in (U_{q_j})^+$, which implies that $\mathcal{A}$ accepts from its state q_j a word v such that $\Theta_{\neq i}(v) = (u_3)^\omega$. By appending a run of $\mathcal{A}$ accepting v from q_j to the path π , we obtain a suitable run ρ . $\square$

We now show that every word accepted by the automata $\mathcal{A}_n$, for all $n \geq 0$, is also accepted by $\mathcal{A}''$.

Theorem 6.2.2. *If an ultimately periodic word w'' is such that for every $n \geq 0$, there exists an accepting run ρ_n of $\mathcal{A}$ such that*

- *the word w_n read by ρ_n satisfies $\Theta_{\neq i}(w_n) = w''$, and*
- *the $(n+1)$ -th transition with a non-empty label followed by ρ_n is associated to the quantified variable x_i ,*

then $w'' \in L(\mathcal{A}'')$.

Proof. For a given value of n , the run ρ_n takes the form

$$\rho_n : q_{n,0} \xrightarrow{a_0} q_{n,1} \xrightarrow{a_1} q_{n,2} \xrightarrow{a_2} \dots q_{n,n} \xrightarrow{\begin{smallmatrix} a_n \\ \{x_i\} \end{smallmatrix}} q_{n,n+1} \xrightarrow{a_{n+1}} \dots,$$

where $q_{n,0}$ is initial, and $w'' = a_0 a_1 a_2 \dots$ with $|a_j| = 1$ for all $j \geq 0$. The $(n+1)$ -th transition with a non-empty label followed by this path is associated

to the quantified variable x_i . Since w'' is ultimately periodic, w_n is ultimately periodic as well. By Lemma 6.2.1, we can assume w.l.o.g. that ρ_n is of the form

$$\rho_n : q_{n,0} \xrightarrow{a_0} \cdots q_{n,n} \xrightarrow{\{x_i\}^{a_n}} q_{n,n+1} \xrightarrow{a_{n+1}} \cdots q_{n,j_n} \xrightarrow{u} q_{n,j_n+\delta_n} \xrightarrow{u} q_{n,j_n+2\delta_n} \xrightarrow{u} \cdots$$

where $j_n \geq n+1$, $\delta_n = |u| > 0$, and $q_{n,j_n} = q_{n,j_n+\delta_n} = q_{n,j_n+2\delta_n} = \cdots$ is an accepting state.

By construction of the automaton $\mathcal{A}'$ (cf. Section 6.2.2), the sets of states $\{q_{0,0}, q_{1,0}, q_{2,0}, \dots\}$, $\{q_{0,1}, q_{1,1}, q_{2,1}, \dots\}$, $\{q_{0,2}, q_{1,2}, q_{2,2}, \dots\}$ correspond (respectively) to the successive states $\bar{q}_0, \bar{q}_1, \bar{q}_2, \dots$ of a run of $\mathcal{A}'$ reading w'' . Since w'' is ultimately periodic, by Lemma 6.2.1, $\mathcal{A}'$ admits a run ρ' of the form

$$\rho' : \bar{q}_0 \xrightarrow{u_1} \bar{q} \xrightarrow{u_2} \bar{q} \xrightarrow{u_3} \bar{q} \xrightarrow{u_3} \cdots,$$

such that $w'' = u_1 \cdot (u_2)^\omega$. Let us pick an arbitrary value of n , say $n = 0$ for the sake of simplicity. The length of u_1 can be freely increased in ρ' , thus we can assume w.l.o.g. that u_1 is such that $|u_1| = j_0 + \ell\delta_0$ for some $\ell \in \mathbb{N}$, which implies $q_{0,j_0+\ell\delta_0} \in \bar{q}$, where $q_{0,j_0+\ell\delta_0}$ is an accepting state of $\mathcal{A}$.

Recall that the run ρ' simulates a set of runs of $\mathcal{A}$ accepting w_n such that $\Theta_{\neq i}(w_n) = w''$. Whenever ρ' visits the state $\bar{q}$, each of these runs visits some state $q_j \in \bar{q}$. It follows that for each $q_j \in \bar{q}$, there exist finite words v_j, t_j and a state $q_k \in \bar{q}$ such that $\mathcal{A}$ admits a run of the form

$$q_j \xrightarrow{v_j} q_k \xrightarrow{t_j} q_k \xrightarrow{t_j} q_k \xrightarrow{t_j} \cdots,$$

with $\Theta_{\neq i}(v_j) = (u_2)^{m_j}$ and $\Theta_{\neq i}(t_j) = (u_2)^{p_j}$ for some $m_j, p_j > 0$.

By defining $p = \text{lcm}(\max_{q_j \in \bar{q}} \{m_j\}, \text{lcm}_{q_j \in \bar{q}} \{p_j\})$, we now have that

- for every $q_j \in \bar{q}$, there exists a state q_k of $\mathcal{A}$ such that this automaton admits paths $q_j \xrightarrow{v} q_k$ and $q_k \xrightarrow{t} q_k$, where v, t are such that $\Theta_{\neq i}(v) = \Theta_{\neq i}(t) = (u_2)^p$,
- the automaton $\mathcal{A}'$ admits a path $\bar{q} \xrightarrow{(u_2)^p} \bar{q}$,
- the set $\bar{q}$ contains at least one accepting state of $\mathcal{A}$.

For those properties, we deduce $(u_2)^p \in U_{\bar{q}}'$. It follows that $((u_2)^p)^\omega = (u_2)^\omega$ is accepted by the widget associated to $\bar{q}$, hence that $\mathcal{A}''$ admits the run

$$\bar{q}_0 \xrightarrow{u_1} \bar{q} \xrightarrow{\varepsilon} q_{\text{repeat}} \xrightarrow{(u_2)^p} q_{\text{repeat}} \xrightarrow{(u_2)^p} q_{\text{repeat}} \xrightarrow{(u_2)^p} \cdots$$

that accepts w'' . □

6.2.7 Complexity

The algorithm introduced in Section 6.2.4 for computing an automaton accepting $\forall x(L(\mathcal{A}_\varphi))$ from an automaton $\mathcal{A}_\varphi$ recognizing the set of models of a formula φ runs in time $2^{O(|\mathcal{A}_\varphi|)}$, where $|\mathcal{A}_\varphi|$ is the size of $\mathcal{A}_\varphi$. This complexity is tight according to the following result.

Theorem 6.2.3. *There exist a family of formulas φ_n for $n \in \mathbb{N}$, and automata $\mathcal{A}^{(n)}$ accepting the encoding of the models of φ_n , such that $|\mathcal{A}^{(n)}| = O(n)$, and the number of states of any automaton accepting $\forall x(L(\mathcal{A}^{(n)}))$ is at least $O(2^n)$.*

Proof. The automaton given in Figure 6.5 recognizes the set of models of

$$\varphi(x_1, x_2, X_1) \equiv x_2 = x_1 + n \wedge X_1(x_1) \Leftrightarrow X_1(x_2),$$

where $n \geq 1$ is a parameter¹.

By removing the variable x_2 from all transition labels of this automaton, one obtains an automaton $\mathcal{A}^{(n)}$ of size $O(n)$ that recognizes the models of $\exists x_2. (x_2 = x_1 + n \wedge X_1(x_1) \Leftrightarrow X_1(x_2))$. One then observes that any automaton recognizing the models of $\forall x_1. \exists x_2. (x_2 = x_1 + n \wedge X_1(x_1) \Leftrightarrow X_1(x_2))$ essentially checks that the value of the predicate X_1 is identical to itself shifted by n positions, in other words, that it is periodic with the period n . Such an automaton must therefore have a memory that can store n bits of information, hence its number of states must at least be equal to 2^n . $\square$

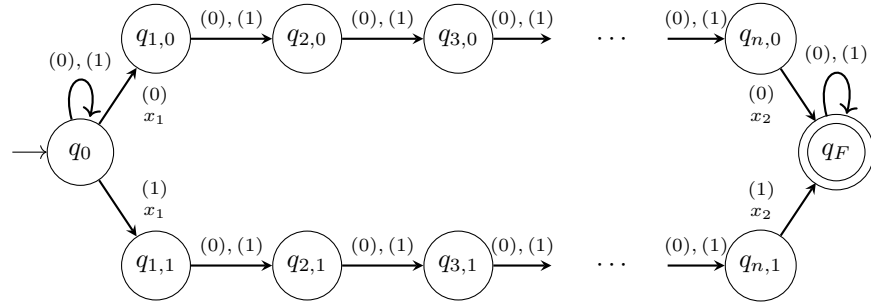


Figure 6.5: Automaton for $x_2 = x_1 + n \wedge X_1(x_1) \Leftrightarrow X_1(x_2)$.

6.3 Towards an Extension for Other Linear Orderings

In this section, we explore some leads on how to tackle first-order universal quantification over different classes of linear orderings. The ultimate goal would be to have a procedure that handles all linear orderings, but this is an ambitious task. Intermediate steps such as covering the set of countable ordinals or countable and scattered linear orderings may already constitute some challenge.

¹The subformula $x_2 = x_1 + n$ is technically not defined in $\text{MFO}(\mathbb{N}, <)$, but can be translated into an equisatisfiable $\text{MFO}(\mathbb{N}, <)$ formula, similarly as what is done in Section 3.1.

6.3.1 Reduction to General Automata Intersection

We saw in Section 6.2 that over $\mathbb{N}$, first-order universal quantification corresponds to computing an infinite intersection of automata $\bigcap_{n \in \mathbb{N}} \mathcal{A}_n$, where each automaton $\mathcal{A}_n$ enforces that the quantified variable takes the value $n \in \mathbb{N}$. The same principle cannot however be directly applied to more general classes of domains. Over $\mathbb{R}$ for instance, there does not exist a finite-state automaton enforcing that a given variable takes a specific value, such as 0, 1, or $\sqrt{2}$.

We now propose an alternate operation on automata on linear orderings which we call *general automata intersection*, which intuitively corresponds to the ability of simulating an arbitrary set of copies of a given automaton over a shared word. The point is that first-order universal quantification reduces to general automata intersection, but the latter may offer a simpler setting to describe the procedure and prove its correctness.

Definition 6.3.1. *Given two automata $\mathcal{A}$ and $\mathcal{B}$ with their respective sets of states $Q_{\mathcal{A}}$ and $Q_{\mathcal{B}}$, we say that $\mathcal{B}$ simulates the general intersection of $\mathcal{A}$ if there exists a function $f : Q_{\mathcal{B}} \rightarrow 2^{Q_{\mathcal{A}}} \setminus \{\emptyset\}$ such that the two following constraints are satisfied:*

(Completeness) *For every non-empty set Π of accepting runs of $\mathcal{A}$ each reading the same word u , there exists an accepting run $\bar{\pi}$ of $\mathcal{B}$ reading u such that, for every cut $c \in \widehat{u}$, one has $f(\bar{\pi}(c)) = \{\pi(c) \mid \pi \in \Pi\}$.*

(Soundness) *For every accepting run $\bar{\pi}$ of $\mathcal{B}$ reading u , there exists a set Π of accepting runs of $\mathcal{A}$ each reading u as well, such that, for every cut $c \in \widehat{u}$, one has $f(\bar{\pi}(c)) = \{\pi(c) \mid \pi \in \Pi\}$.*

Notice that this definition implies that any automaton simulating the general intersection of $\mathcal{A}$ necessarily accepts the exact same language as $\mathcal{A}$. This is intentional as we are not interested in performing an operation on the language accepted by $\mathcal{A}$, but rather gain information about how different runs can be synchronized over a shared word. Furthermore, let us explicitly mention that we do not know if, for any given automaton on linear orderings $\mathcal{A}$, there exists an automaton $\mathcal{B}$ that simulates the general intersection of $\mathcal{A}$.

In the following, we assume that we have access to a procedure that, for any input automaton on linear orderings $\mathcal{A}$, computes an automaton $\tilde{\cap}(\mathcal{A})$ that simulates the general intersection of $\mathcal{A}$. We now show how first-order universal quantification then reduces to general automata intersection.

Consider an (n, m) -automaton on linear orderings $\mathcal{A}$, and let $i \in [1, n]$. Then, for every word u that belongs to $\forall x_i(L(\mathcal{A}))$, there exists a corresponding set of accepting runs of $\mathcal{A}$. However, unlike in the case of the general intersection, all these runs do not all read the same word, indeed, the position of x_i is different for each run so as to cover every single position in $|u|$. But the words read from the other $n - 1 + m$ tapes, i.e., the tapes that are not associated with x_i , must be identical.

A first-order universal quantification algorithm, taking as input an (n, m) -automaton $\mathcal{A} = (Q, \{0, 1\}^{n+m}, I, F, \Delta^s, \Delta^\ell)$ and an index $i \in [1, n]$, can be obtained by performing the following steps.

1. We start by creating three distinct copies of each state $q \in Q$, denoted by $q_{[x_i]}$, $q_{[\neg x_i]}$, and q_{lim} , respectively designed to discriminate on the basis of whether the state q is visited immediately after following a successor transition associated with the variable x_i , a successor transition not associated with x_i , or is reached via a left-limit transition. The state q_{lim} will also serve for when the state q is an initial state visited at the first cut. This yields an alternate set of states Q_{bis} defined as follows:

$$Q_{\text{bis}} = \bigcup_{q \in Q} \{q_{[x_i]}, q_{[\neg x_i]}, q_{\text{lim}}\}.$$

2. We then compute the set of successor transitions Δ_{bis}^s derived from Δ^s so as to enforce the semantics of the states $q_{[x_i]}$, $q_{[\neg x_i]}$, and q_{lim} introduced at the previous steps. This updated set of successor transitions is defined as follows:

$$\begin{aligned} \Delta_{\text{bis}}^s = & \{(p_{[\neg x_i]}, \alpha, q_{[x_i]}) \mid (p, \alpha, q) \in \Delta^s \wedge \Theta_{=i}(\alpha) = 1\} \\ & \cup \{(p_{[\neg x_i]}, \alpha, q_{[\neg x_i]}) \mid (p, \alpha, q) \in \Delta^s \wedge \Theta_{=i}(\alpha) = 0\} \\ & \cup \{(p_{[x_i]}, \alpha, q_{[\neg x_i]}) \mid (p, \alpha, q) \in \Delta^s \wedge \Theta_{=i}(\alpha) = 0\} \\ & \cup \{(p_{\text{lim}}, \alpha, q_{[x_i]}) \mid (p, \alpha, q) \in \Delta^s \wedge \Theta_{=i}(\alpha) = 1\} \\ & \cup \{(p_{\text{lim}}, \alpha, q_{[\neg x_i]}) \mid (p, \alpha, q) \in \Delta^s \wedge \Theta_{=i}(\alpha) = 1\}. \end{aligned}$$

3. Then, we proceed to the computation of an analogous set of limit transitions Δ_{bis}^ℓ enforcing the semantics of the states $q_{[x_i]}$, $q_{[\neg x_i]}$, and q_{lim} with respect to limit transitions. This updated set of limit transitions is defined as follows:

$$\begin{aligned} \Delta_{\text{bis}}^\ell = & \{P_{\text{bis}} \rightarrow q_{\text{lim}} \mid P_{\text{bis}} \subseteq Q_{\text{bis}} \wedge \exists P \subseteq Q. P \rightarrow q \in \Delta^\ell \\ & \wedge \forall p \in P. P_{\text{bis}} \cap \{p_{[\neg x_i]}, p_{\text{lim}}\} \neq \emptyset \\ & \wedge \forall p_{[\neg x_i]} \in P_{\text{bis}}. p \in P \wedge \forall p_{\text{lim}} \in P_{\text{bis}}. p \in P\} \\ & \cup \{q_{\bowtie} \rightarrow P_{\text{bis}} \mid \exists q \in Q, P \subseteq Q. q_{\bowtie} \in \{q_{[x_i]}, q_{[\neg x_i]}, q_{\text{lim}}\} \wedge q \rightarrow P \in \Delta^\ell \\ & \wedge \forall p \in P. P_{\text{bis}} \cap \{p_{[\neg x_i]}, p_{\text{lim}}\} \neq \emptyset \\ & \wedge \forall p_{[\neg x_i]} \in P_{\text{bis}}. p \in P \wedge \forall p_{\text{lim}} \in P_{\text{bis}}. p \in P\}. \end{aligned}$$

Intuitively, the targets of left-limit transitions can only be states of the form q_{lim} , and the limit sets of left and right-limit transitions must cover the cases where, for a given state $p \in Q$, the variant states actually relevant to the limit set is either $p_{[\neg x_i]}$, p_{lim} , or both. Notice that the limit sets we introduce never contain states of the form $q_{[x_i]}$. This is because, as a direct consequence of the structure of (n, m) -automata that only admit accepting runs visiting exactly one transition labeled with x_i , these states can only be visited once per accepting run.

4. Next, we combine the previous definitions with the set I_{bis} of initial states defined as

$$I_{\text{bis}} = \{q_{\text{lim}} \mid q \in I\},$$

and the set F_{bis} of accepting states defined as

$$F_{\text{bis}} = \{q_{\bowtie} \in Q_{\text{bis}} \mid \exists q \in F. q_{\bowtie} \in \{q_{[x_i]}, q_{[\neg x_i]}, q_{\text{lim}}\}\},$$

and obtain the automaton $\mathcal{A}_{\text{bis}} = (Q_{\text{bis}}, \{0, 1\}^{n+m}, I_{\text{bis}}, F_{\text{bis}}, \Delta_{\text{bis}}^s, \Delta_{\text{bis}}^\ell)$.

5. We then remove the tape associated with the first-order variable x_i from the automaton $\mathcal{A}_{\text{bis}}$ altogether. Formally, this simply consists in replacing each successor transition of the form (q, α, p) by $(q, \Theta_{\neq i}(\alpha), p)$. We denote by $\Theta_{\neq i}(\mathcal{A}_{\text{bis}})$ the automaton obtained after removing the i -th tape.
6. The next step consists in computing $\bigcap^\infty (\Theta_{\neq i}(\mathcal{A}_{\text{bis}}))$, i.e., an automaton that simulates the general intersection of the automaton $\Theta_{\neq i}(\mathcal{A}_{\text{bis}})$. We assume we also have access to the function f that maps the states of $\bigcap^\infty (\Theta_{\neq i}(\mathcal{A}_{\text{bis}}))$ to subsets of Q_{bis} . This step depends entirely on the assumption that we have an algorithm to compute an automaton simulating the general intersection of any (n, m) -automata.
7. The last step consists in building a final automaton obtained by removing some states from $\bigcap^\infty (\Theta_{\neq i}(\mathcal{A}'))$. Namely, we only keep the states $\bar{q}$ in $\bigcap^\infty (\Theta_{\neq i}(\mathcal{A}'))$ such that either
- there exists a unique state of the form $q_{[x_i]}$ in $f(\bar{q})$, or
 - every state in $f(\bar{q})$ is of the form q_{lim} .

This ensures that an accepting run $\bar{\pi}$ of $\mathcal{A}_{\forall x_i}$ that reads the word u , corresponds to a set of accepting runs of $\mathcal{A}$ such that, for each $j \in |u|$, at least one of these runs reads the variable x_i at position j .

We call $\mathcal{A}_{\forall x_i}$ the automaton constructed by this procedure.

Lemma 6.3.1. *Provided that general intersection of automata can be computed, the automaton $\mathcal{A}_{\forall x_i}$ recognizes the language $\forall x_i(L(\mathcal{A}))$.*

Proof. Let us start by showing that any word $u \in L(\mathcal{A}_{\forall x_i})$ also belongs to $\forall x_i(L(\mathcal{A}))$. If u belongs to $L(\mathcal{A}_{\forall x_i})$, i.e., there exists an accepting run $\bar{\pi}$ of $\mathcal{A}_{\forall x_i}$ that reads u , it means that there exists a set Π of accepting runs of $\Theta_{\neq i}(\mathcal{A}_{\text{bis}})$, each reading the word u , and such that every successor transition followed by the run $\bar{\pi}$ is of the form $(\bar{q}, \alpha, \bar{q}')$ with $\bar{q}'$ containing a state of the form $q_{[x_i]}$. Because $\mathcal{A}$ (and therefore also $\mathcal{A}_{\text{bis}}$) is an (n, m) -automaton, we know that any accepting run of $\mathcal{A}$ follows a successor transition associated with x_i exactly once. Although the variable x_i no longer appears in the automaton $\Theta_{\neq i}(\mathcal{A}_{\text{bis}})$, a similar property remains true w.r.t. the successor transitions formerly (i.e., in $\mathcal{A}$) associated with x_i . And by construction of the automaton $\mathcal{A}_{\text{bis}}$, this

property is somewhat transferred to the set of successor transitions of the form $(p, \alpha, q_{[x_i]})$. It follows that the set of runs Π can be straightforwardly adapted into a set Π' of accepting runs in $\mathcal{A}$ such that, for all $j \in |u|$, there exists a run $\pi' \in \Pi'$ that reads the word $u \oplus [x_i = j]$. Therefore, u belongs to $\forall x_i(L(\mathcal{A}))$.

To deal with the other direction, let us show that any word $u \in \forall x_i(L(\mathcal{A}))$ is accepted by $\mathcal{A}_{\forall x_i}$. By definition of $\forall x_i(L(\mathcal{A}))$, there necessarily exists a set $\Pi = (\pi_j)_{j \in |u|}$ of $|u|$ accepting runs in $\mathcal{A}$ such that, for all $j \in |u|$, the run π_j reads the word $u \oplus [x_i = j]$. Each of the runs in Π can straightforwardly be adapted into a run in the automaton $\Theta_{\neq i}(\mathcal{A}_{\text{bis}})$ reading u , and the set of the adapted runs from Π can collectively be simulated by a run $\bar{\pi}$ in the automaton $\bigcap^\infty(\Theta_{\neq i}(\mathcal{A}_{\text{bis}}))$. Furthermore, this set of adapted runs is such that, for any pair of consecutive cuts (c, c') , the set of successor transitions collectively followed to go from c to c' contains one successor transition initially associated with the variable x_i in $\mathcal{A}$, or equivalently, one of the adapted runs in $\Theta_{\neq i}(\mathcal{A}_{\text{bis}})$ follows a successor transition of the form $(p, \alpha, q_{[x_i]})$ to go from the cut c to the cut c' . This is enough to ensure that all the states visited by the run $\bar{\pi}$ in $\bigcap^\infty(\Theta_{\neq i}(\mathcal{A}))$ are still present in the automaton $\mathcal{A}_{\forall x_i}$. It follows that $\bar{\pi}$ is an accepting run of $\mathcal{A}_{\forall x_i}$, which concludes the proof. $\square$

It follows that if there exists a procedure that computes, for any (n, m) -automaton $\mathcal{A}$, an automaton $\bigcap^\infty(\mathcal{A})$ that simulates the general intersection of $\mathcal{A}$, then one can straightforwardly obtain a procedure to compute the first-order universal quantification of (n, m) -languages.

6.3.2 Hindrances

While we believe that the notion of general intersection may be a helpful proxy to tackle first-order universal quantification, we would also like to highlight some challenging hurdles to the design of a general intersection procedure.

If one manages to solve general intersection over all countable ordinals, it is safe to say that the case of all countable reverse ordinals could be handled analogously, by mirroring the mechanisms introduced for countable ordinals. It is already a riskier stretch to assume that combining these hypothetical mechanisms for normal and reverse countable ordinals could naturally lead to a solution for the case of all countable scattered linear orderings. It is however completely unclear to us whether such a procedure would also be able to natively handle shuffle structures, or if some specific additional work would be necessary to deal with non-scattered linear orderings.

Another major difficulty we stumbled upon when trying to design algorithms for universal quantification and general intersection, or prove their correctness, is dealing with orderings that admit suborderings of uncountable coinitiality or cofinality. Over such orderings, it is far from trivial (at least for us) to precisely capture the joint behavior of multiple runs in a given automaton on linear orderings.

Notice however that the domains $\mathbb{R}$ and $\mathbb{Q}$ do not possess suborderings of coinitiality or cofinality ω_1 or greater (cf. Property 1.2.1). Therefore, a partial procedure for first-order universal quantification that handles density but leaves out uncountable coinitiality and cofinality would be enough to yield an effective decision procedure for $\text{MFO}(\mathbb{R}, <)$ and $\text{MFO}(\mathbb{Q}, <)$.

Conclusion

In this thesis, we studied the decidability of theories combining arithmetic and uninterpreted symbols. This led us to focus on the monadic first-order theory of order and some of its extensions, for which we investigated the design of effective decision procedures.

In Chapter 3, we narrowed the frontier of decidability when difference logic is combined with uninterpreted symbols. Namely, we showed that difference logic over the reals with a single unary uninterpreted predicate is undecidable, while the monadic first-order theory of order over the reals, with difference constraints restricted to integer variable is decidable. We also provided a translation from this decidable fragment to the monadic first-order theory of order interpreted of the real numbers, for which we developed an alternate self-contained proof of its decidability. In Chapter 4, we lay out a candidate structure for an automata-based decision procedure for the monadic first-order theory of order. Our approach does not discriminate with respect to the intended domains of interpretation up until the last step. In other words, the sketched procedure first builds an automaton accepting the models of an input formula over the class of all linear orderings, and then performs a domain-specific non-emptiness test to check if the automaton accepts a word indexed by a suitable linear ordering. In Chapter 5, we developed non-emptiness tests tailored for the orderings of the reals and rationals. Finally, in Chapter 6, we presented an algorithm to compute the effect of first-order universal quantification on Büchi automata, before proposing an alternate operation — designed to explicitly represent all the different ways an automaton can read a given word — that may offer a more convenient framework to solve first-order universal quantification for automata on linear orderings.

A natural follow-up to this work would be to solve first-order universal quantification for other domains, in particular for $\mathbb{R}$ and $\mathbb{Q}$. An important related question is whether or not automata on linear orderings are closed under first-order universal quantification over the class of all orderings, and if it is not the case, is there a class strictly greater than that of all countable and scattered linear orderings for which this closure property holds.

As another lead towards solving first-order universal quantification, it may be worth investigating a notion analogous to the one of ultimately periodic words, defined over ω -words, which helped prove the correctness of the first-order universal quantification algorithm for Büchi automata, introduced in Section 6.2.

A first step in that direction is presented in [17], where the authors introduce a notion of constructible words over the class of all scattered linear orderings.

Additionally, in order to fulfill the endgoal to obtain effective implementations of decision procedures for $\text{MFO}(<)$ over various domains, the resolved parts of the procedure sketched in this thesis could be further improved and optimized. Some work in that direction have already been done in [18] where the authors propose an optimized version of the non-emptiness test for automata on linear orderings, and in [34] where the problem of efficiently computing the intersection of automata on linear orderings is investigated.

Bibliography

- [1] Ondrej Alexaj, Vojtěch Havlena, Lukáš Holík, Ondřej Lengál, Yong Li, and Nicolas Mazzocchi. Kofola 1.0: A modular approach to ω -regular complementation and inclusion checking. In Eva Darulova, Anthony W. Lin, and Philipp Rümmer, editors, *Computer Aided Verification*, pages 181–197, Cham, 2026. Springer Nature Switzerland.
- [2] Clark Barrett, Daniel Kroening, and Thomas Melham. Problem solving for the 21st century: efficient solvers for Satisfiability Modulo Theories. Technical Report 3, London Mathematical Society and Smith Institute for Industrial Mathematics and System Engineering, June 2014. Knowledge Transfer Report.
- [3] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, Amsterdam, 2009.
- [4] Nicolas Bedon. Finite automata and ordinals. *Theoretical Computer Science*, 156(1):119–144, 1996.
- [5] Nicolas Bedon, Alexis Bès, Olivier Carton, and Chloe Rispal. Logic and rational languages of words indexed by linear orderings. *Theory of Computing Systems*, 46:737–760, 2010.
- [6] Valérie Berthé, Toghrul Karimov, Joris Nieuwveld, Joël Ouaknine, Mihir Vahanwala, and James Worrell. On the decidability of monadic second-order logic with arithmetic predicates. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–14, 2024.
- [7] Alexis Bès. A survey of arithmetical definability. *A tribute to Maurice Boffa*, pages 1–54, 2002.
- [8] Alexis Bès and Olivier Carton. A Kleene theorem for languages of words indexed by linear orderings. *International Journal of Foundations of Computer Science*, 17(03):519–541, 2006.

- [9] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Handbook of satisfiability*, 185(99):457–481, 2009.
- [10] Bernard Boigelot, Thomas Braipson, and Tom Clara. Epsilon automata on linear orderings. In *International Conference on Implementation and Application of Automata*, pages 41–54. Springer, 2025.
- [11] Bernard Boigelot, Pascal Fontaine, and Baptiste Vergain. Decidability of difference logic over the reals with uninterpreted unary predicates. In *International Conference on Automated Deduction*, pages 542–559. Springer, 2023.
- [12] Bernard Boigelot, Pascal Fontaine, and Baptiste Vergain. Deciding reachability in automata on words indexed by the reals and rationals. *Theoretical Computer Science*, 1080:116048, 2026.
- [13] Bernard Boigelot, Pascal Fontaine, and Baptiste Vergain. On the theory of order over real numbers. *Mathematics in Computer Science*, 20(1):5, 2026.
- [14] Bernard Boigelot, Pascal Fontaine, and Baptiste Vergain. Universal first-order quantification over automata. *International Journal of Foundations of Computer Science*, 37(01):5–21, 2026.
- [15] Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Transactions on Computational Logic (TOCL)*, 6(3):614–633, 2005.
- [16] Bernard Boigelot and Louis Latour. Counting the solutions of Presburger equations without enumerating them. *Theoretical Computer Science*, 313(1):17–29, 2004.
- [17] Thomas Braipson and Tom Clara. Constructible Words Characterize Rational Languages of Words Indexed by Scattered Linear Orderings. In Michal Koucký and Daniela Petrişan, editors, *51st International Symposium on Mathematical Foundations of Computer Science (MFCS 2026)*, volume 386 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:18, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [18] Thomas Braipson and Tom Clara. Efficiently testing emptiness of automata on linear orderings. In Taylor J. Smith, editor, *Implementation and Application of Automata*, pages 1–15, Cham, 2026. Springer Nature Switzerland.
- [19] Christopher W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *ACM Sigsam Bulletin*, 37(4):97–108, 2003.

- [20] Véronique Bruyère and Olivier Carton. Automata on linear orderings. *Journal of Computer and System Sciences*, 73(1):1–24, 2007.
- [21] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.
- [22] J. Richard Büchi. On a decision method in restricted second order arithmetic. In *Proc. Intl. Congress on Logic, Methodology and Philosophy of Science*, 1962.
- [23] J. Richard Büchi. Transfinite automata recursions and weak second order theory of ordinals. In *Proc. Intl. Congress on Logic, Methodology, and Philosophy of Science*, 1965.
- [24] J. Richard Büchi and Charles Zaiontz. *Deterministic Automata and the Monadic Theory of Ordinals $< \omega_2$* , pages 493–516. Springer New York, 1990.
- [25] John P. Burgess and Yuri Gurevich. The decision problem for linear temporal logic. *Notre Dame Journal of Formal Logic*, 26(2):115–128, 1985.
- [26] Hugues Calbrix, Maurice Nivat, and Andreas Podelski. Ultimately periodic words of rational w -languages. In *Proc. 9th MFPS*, volume 802 of *LNCS*, pages 554–566. Springer, 1993.
- [27] Georg Cantor. Über unendliche, lineare Punktmannichfaltigkeiten. *Mathematische Annalen*, 20(1):113–121, 1882.
- [28] Georg Cantor. Beiträge zur Begründung der transfiniten Mengenlehre. *Mathematische Annalen*, 46:481–512, 1895.
- [29] Olivier Carton. Accessibility in automata on scattered linear orderings. In *Proc. Intl. Symp. on Mathematical Foundations of Computer Science*, pages 155–164. Springer, 2002.
- [30] Olivier Carton, Thomas Colcombet, and Gabriele Puppis. An algebraic approach to MSO-definability on countable linear orderings. *The Journal of Symbolic Logic*, 83(3):1147–1189, 2018.
- [31] Bobby F. Caviness. On canonical forms and simplification. *Journal of the ACM (JACM)*, 17(2):385–396, 1970.
- [32] Yaacov Choueka. Finite automata, definable sets, and regular expressions over ω^n -tapes. *Journal of Computer and System Sciences*, 17(1):81–97, 1978.
- [33] Alonzo Church. An unsolvable problem of elementary number theory. *American journal of mathematics*, 58(2):345–363, 1936.

- [34] Tom Clara. Efficient representation and manipulation of automata on linear orderings. Master's thesis, Université de Liège, <http://hdl.handle.net/2268.2/23281>, 2025.
- [35] Paul J. Cohen. The independence of the continuum hypothesis. *Proceedings of the National Academy of Sciences*, 50(6):1143–1148, 1963.
- [36] Thomas Colcombet. Factorization forests for infinite words and applications to countable scattered linear orderings. *Theoretical Computer Science*, 411(4-5):751–764, 2010.
- [37] Thomas Colcombet and Alexander Rabinovich. On the expansion of monadic second-order logic with Cantor-Bendixson rank and order type predicates. In *Computer Science Logic (CSL)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025.
- [38] Julien Cristau. Automata and temporal logic over arbitrary linear time. In *Proc. Intl. Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 4 of *LIPICs*, pages 133–144, 2009.
- [39] Julien Cristau. *Jeux et automates sur les ordres*. PhD thesis, Université Paris Diderot, 2010.
- [40] David Detlefs, Greg Nelson, and James B Saxe. Simplify: a theorem prover for program checking. *Journal of the ACM (JACM)*, 52(3):365–473, 2005.
- [41] Emmanuele DiBenedetto. *Real analysis*. Springer, 2002.
- [42] Peter J. Downey. Undecidability of Presburger arithmetic with a single monadic predicate letter. Technical report, Center for Research in Computer Technology, Harvard University, 1972.
- [43] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer, Berlin, Heidelberg, 1995.
- [44] Andrzej Ehrenfeucht. Decidability of the theory of linear order. *Notices of the Am. Math. Soc.*, 6:268–269, 1959.
- [45] Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fundamenta Mathematicae*, 49:129–141, 1961.
- [46] Calvin C. Elgot and Michael O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *The Journal of Symbolic Logic*, 31(2):169–181, 1966.
- [47] E. Allen Emerson and Joseph Y. Halpern. “Sometimes” and “not never” revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986.
- [48] Herbert B. Enderton. *A mathematical introduction to logic (2nd edition)*. Academic Press, Boston, 2001.

- [49] Solomon Feferman and Robert Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.
- [50] Abraham A. Fraenkel, Yehoshua Bar-Hillel, and Azriel Levy. *Foundations of set theory*, volume 67. Elsevier, 1973.
- [51] Roland Fraïssé. *Sur quelques classifications des systemes de relations*. PhD thesis, Imprimerie Durand, 1955.
- [52] Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*. Oxford University Press, 07 1994.
- [53] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 163–173, 1980.
- [54] Vijay Ganesh and Moshe Y Vardi. On the unreasonable effectiveness of SAT solvers, 2020.
- [55] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.
- [56] Kurt Gödel. *The consistency of the axiom of choice and of the generalized continuum-hypothesis with the axioms of set theory*. Princeton University Press, 1940.
- [57] Yuri Gurevich, Menachem Magidor, and Saharon Shelah. The monadic theory of ω_2 . *The Journal of Symbolic Logic*, 48(2):387–398, 1983.
- [58] Yuri Gurevich and Alexander Rabinovich. Definability and undefinability with real order at the background. *The Journal of Symbolic Logic*, 65(2):946–958, 2000.
- [59] Yuri Gurevich and Saharon Shelah. Monadic theory of order and topology in ZFC. *Annals of Mathematical Logic*, 23(2-3):179–198, 1982.
- [60] Christoph Haase. A survival guide to Presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, 2018.
- [61] Peter Habermehl, Michal Hecko, Vojtech Havlena, Lukáš Holík, and Ondřej Lengál. Algebraic reasoning meets automata in solving linear integer arithmetic. In *Proc. Intl. Conf. on Computer Aided Verification (CAV)*, volume 14681 of *Lecture Notes in Computer Science*, pages 42–67, Berlin, Heidelberg, 2024. Springer.
- [62] Joseph Y. Halpern. Presburger arithmetic with unary predicates is Π_1^1 complete. *The Journal of Symbolic Logic*, 56(2):637–642, June 1991.

- [63] Philipp Hieronymi. When is scalar multiplication decidable? *Annals of Pure and Applied Logic*, 170(10):1162–1175, 2019.
- [64] Jaakko Hintikka. Distributive normal forms in the calculus of predicates. *Acta philosophica Fennica, fasc. 6*, 1953.
- [65] Yoram Hirshfeld and Alexander Rabinovich. Logics for real time: Decidability and complexity. *Fundamenta Informaticae*, 62(1):1–28, 2004.
- [66] Matthias Horbach, Marco Voigt, and Christoph Weidenbach. The universal fragment of Presburger arithmetic with unary uninterpreted predicates is undecidable. *arXiv preprint arXiv:1703.01212*, 2017.
- [67] Johan A. W. Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.
- [68] Richard Kaye. *Models of Peano arithmetic*. Oxford University Press, 1991.
- [69] Nils Klarlund. Mona & Fido: The logic-automaton connection in practice. In *International Workshop on Computer Science Logic*, pages 311–326. Springer, 1997.
- [70] Stephen Cole Kleene. *Representation of events in nerve nets and finite automata*, volume 34. Princeton University Press, 1956.
- [71] Orna Kupferman and Moshe Y. Vardi. Complementation constructions for nondeterministic automata on infinite words. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 206–221. Springer, 2005.
- [72] Hans Läuchli. A decision procedure for the weak second order theory of linear order. In *Studies in Logic and the Foundations of Mathematics*, volume 50, pages 189–197. Elsevier, 1968.
- [73] Hans Läuchli and Jon Leonard. On the elementary theory of linear order. *Fundamenta Mathematicae*, 59(1):109–116, 1966.
- [74] Angus Macintyre and A. J. Wilkie. On the decidability of the real exponential field. *Kreisel’s Mathematics*, 115:451, 1996.
- [75] Johann A. Makowsky. Algorithmic uses of the Feferman–Vaught theorem. *Annals of Pure and Applied Logic*, 126(1-3):159–213, 2004.
- [76] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems*. Springer, 1992.
- [77] Sven Manthe. The Borel monadic theory of order is decidable. *arXiv preprint arXiv:2410.00887*, 2024.
- [78] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and control*, 9(5):521–530, 1966.

- [79] Robert McNaughton and Hisao Yamada. Regular expressions and state graphs for automata. *IRE transactions on Electronic Computers*, EC-9(1):39–47, 1960.
- [80] Albert R. Meyer. Weak monadic second order theory of successor is not elementary-recursive. In *Logic Colloquium: Symposium on Logic Held at Boston, 1972–73*, pages 132–154. Springer, 1972.
- [81] Andrzej Mostowski. On direct products of theories. *The Journal of Symbolic Logic*, 17(1):1–31, 1952.
- [82] Reed Oei, Dun Ma, Christian Schulz, and Philipp Hieronymi. Pecan: An automated theorem prover for automatic sequences using Büchi automata. *arXiv preprint arXiv:2102.01727*, 2021.
- [83] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [84] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, pages 46–57. IEEE, 1977.
- [85] Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchen die Addition als einzige Operation hervortritt. In *Comptes-Rendus du 1er Congrès des Mathématiciens des Pays Slaves*, 1929.
- [86] Arthur N. Prior. *Time and Modality*. Oxford University Press, London, 1957.
- [87] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.
- [88] Michael O. Rabin. Decidable theories. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 595–629. Elsevier, 1977.
- [89] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959.
- [90] Alexander Rabinovich. Temporal logics over linear time domains are in PSPACE. *Information and Computation*, 210:40–67, 2012.
- [91] Alexander Rabinovich. A proof of Kamp’s theorem. *Logical Methods in Computer Science*, 10, 2014.
- [92] Frank P. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 30:264–286, 1930.

- [93] Mark Reynolds. The complexity of the temporal logic with “until” over general linear time. *Journal of Computer and System Sciences*, 66(2):393–426, 2003.
- [94] Mark Reynolds. The complexity of decision problems for linear temporal logics. *Journal of Studies in Logic*, 3(1):19–50, 2010.
- [95] Mark Reynolds. The complexity of temporal logic over the reals. *Annals of Pure and Applied Logic*, 161(8):1063–1096, 2010.
- [96] Daniel Richardson. Some undecidable problems involving elementary functions of a real variable. *The Journal of Symbolic Logic*, 33(4):514–520, 1969.
- [97] Chloé Rispal. *Automates sur les ordres linéaires: Complémentation*. PhD thesis, Université de Marne la Vallée, 2004.
- [98] Chloé Rispal and Olivier Carton. Complementation of rational sets on countable scattered linear orderings. *International Journal of Foundations of Computer Science*, 16(04):767–786, 2005.
- [99] Raphael M. Robinson. Restricted set-theoretical definitions in arithmetic. *Proceedings of the American Mathematical Society*, 9(2):238–242, 1958.
- [100] Joseph G. Rosenstein. *Linear orderings*. Academic press, 1982.
- [101] Bertrand Russel. *The Principles of Mathematics*. Cambridge University Press, 1903.
- [102] Shmuel Safra. On the complexity of ω -automata. In *Proc. 29th Annual Symposium on Foundations of Computer Science*, pages 319–327. IEEE, 1988.
- [103] Jeffrey Shallit. *The Logical Approach to Automatic Sequences*, volume 482. Cambridge University Press, 2022.
- [104] Claude E. Shannon. A universal Turing machine with two internal states. *Automata studies*, 34:157–165, 1956.
- [105] Saharon Shelah. The monadic theory of order. *Annals of Mathematics*, 102(3):379–419, 1975.
- [106] Wacław Sierpiński. *Cardinal and ordinal numbers (2nd edition)*. PWN, Warszawa, 1965.
- [107] Micheal Sipser. *Introduction to the Theory of Computation*. Cengage Learning, third edition, 2012.
- [108] A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49(2-3):217–237, 1987.

- [109] Thoralf Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Bewiesbarkeit mathematischer Sätze nebst einem Theorem über dichte Mengen. 1920.
- [110] Thoralf Skolem. *Über einige Satzfunktionen in der Arithmetik*. Skrifter utgitt av det Norske Videnskaps-Akademi i Oslo, 1, Mat-Nat. 1931.
- [111] Larry J. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, Massachusetts Institute of Technology, 1974.
- [112] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Univ. of California Press, 2nd edition, 1951.
- [113] Wolfgang Thomas. Automata on infinite objects. In *Formal Models and Semantics*, pages 133–191. Elsevier, 1990.
- [114] Wolfgang Thomas. Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In *Structures in Logic and Computer Science*, pages 118–143. Springer, 1997.
- [115] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of formal languages*, pages 389–455. Springer, 1997.
- [116] Boris A. Trakhtenbrot. Finite automata and the logic of one-place predicates. *Sibirskii Matematicheskii Zhurnal*, 3(1):103–131, 1962.
- [117] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42:230–265, 1936.
- [118] Moshe Y. Vardi. The Büchi complementation saga. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 12–22. Springer, 2007.
- [119] Paul S. Wang. The undecidability of the existence of zeros of real elementary functions. *Journal of the ACM (JACM)*, 21(4):586–589, 1974.
- [120] A. J. Wilkie. *Schanuel’s Conjecture and the Decidability of the Real Exponential Field*, pages 223–230. Springer Netherlands, Dordrecht, 1997.
- [121] Jerzy Wojciechowski. Finite automata on transfinite sequences and regular expressions. *Fundamenta Informaticae*, 8(3-4):379–396, 1985.
- [122] Ruth S. Zeitman. *The composition method*. PhD thesis, Wayne State University, 1994.