

Universal First-Order Quantification over Automata

Bernard Boigelot, Pascal Fontaine, and Baptiste Vergain

Montefiore Institute, University of Liège, B28
4000 Liège, Belgium

{bernard.boigelot, pascal.fontaine, bvergain}@uliege.be

Abstract

Deciding formulas that mix arithmetic and uninterpreted predicates is of practical interest, notably for applications in verification. One approach for obtaining a decision procedure consists in building by structural induction an automaton that recognizes the set of models of the formula under analysis, and then testing whether this automaton accepts a non-empty language. A drawback is that universal quantification is usually handled by a reduction to existential quantification and complementation. For logical formalisms in which models are encoded as infinite words, this hinders the practical use of this method due to the difficulty of complementing infinite-word automata. The contribution of this paper is to introduce an algorithm for directly computing the effect of universal first-order quantifiers on automata recognizing sets of models, for formulas involving natural numbers encoded in unary notation. This paves the way to implementable decision procedures for various arithmetic theories.

Keywords: Infinite-word automata, first-order logic, quantifier elimination, satisfiability

1 Introduction

Automated reasoning with arithmetic theories is of primary importance, notably for verification, where Satisfiability Modulo Theories (SMT) solvers are regularly used to discharge proof obligations. It is well known however that mixing arithmetic with uninterpreted symbols quickly leads to undecidability. For instance, extending Presburger arithmetic, i.e., the first-order additive theory of integer numbers, with just one uninterpreted unary predicate makes it undecidable [8, 9, 16]. There exist decidable fragments mixing arithmetic and uninterpreted symbols that are expressive enough to be interesting, for instance, the monadic second-order theory of $\mathbb{N}$ under one successor (S1S).

The decidability of S1S has been established thanks to the concept of infinite-word automaton [6]. In order to decide whether a formula φ is satisfiable, the approach consists in building an automaton that recognizes the set of its models, encoded in a suitable way, and then checking that this automaton accepts a non-empty language. Such an automaton has one separate input tape for each first-order and second-order free variable of φ . 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. For instance, applying an existential quantifier simply amounts to removing from the automaton the input tape associated to the quantified variable. Universal quantification reduces to existential quantification thanks to the equivalence $\forall x \varphi \equiv \neg \exists x \neg \varphi$.

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 [10] uses this method to decide a restricted version of S1S, and tools such as LASH [3] and Shasta [15] use a similar technique to decide Presburger arithmetic. LASH also generalizes this result by providing an implemented decision procedure for the first-order additive theory of mixed integer and real variables [2].

A major issue in practice is that the elimination of universal quantifiers relies on complementation, which is an operation that is not easily implemented for infinite-word automata [14, 18]. 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 only handles *Weak* S1S (WS1S) which is, schematically, a restriction of S1S to finite subsets of natural numbers [5]. The tool LASH 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 [2].

The contribution of this paper is to introduce a direct algorithm for computing the effect of universal first-order quantification over infinite-word automata. This is an essential step towards practical decision procedures for more expressive fragments mixing arithmetic with uninterpreted symbols. The considered automata are those that recognize models of formulas over natural numbers encoded in unary notation. This algorithm does not rely on complementation, and can be implemented straightforwardly on unrestricted infinite-word automata. As an example of its potential applications, this algorithm leads to a practically implementable decision procedure for the first-order theory of natural numbers with the order relation and uninterpreted unary predicates. It also paves the way to a decision procedure for SMT solvers for the UFIDL (Uninterpreted Functions and Integer Difference Logic) logic with only unary predicates.

2 Basic notions

2.1 Logic

We address the problem of deciding satisfiability for formulas expressed in first-order structures of the form $(\mathbb{N}, R_1, R_2, \dots)$, where $\mathbb{N}$ is the *domain* of natural numbers, and $R_1, R_2, \dots$ are (interpreted) *relations* over tuples of values in $\mathbb{N}$. More precisely, each R_i is defined as a relation $R_i \subseteq \mathbb{N}^{\alpha_i}$ for some $\alpha_i > 0$ called the *arity* of R_i .

The formulas in such a structure involve *first-order variables* $x_1, x_2, \dots$, and *second-order variables* $X_1, X_2, \dots$. Formulas are recursively defined as

- $\top, \perp, x_i = x_j, X_i = X_j, X_i(x_j)$ or $R_i(x_{j_1}, \dots, x_{j_{\alpha_i}})$, where $i, j, j_1, j_2, \dots \in \mathbb{N}_{>0}$ (*atomic formulas*),
- $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2$ or $\neg \varphi$, where φ_1, φ_2 and φ are formulas, or
- $\exists x_i \varphi$ or $\forall x_i \varphi$, where $i \in \mathbb{N}_{>0}$ and φ is a formula.

We write $\varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$ to express that $x_1, \dots, x_k, X_1, \dots, X_\ell$ are the free variables of φ , i.e., that φ does not involve other unquantified variables.

An *interpretation* I for a formula $\varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$ is an assignment of values $I(x_i) \in \mathbb{N}$ for all $i \in [1, k]$ and $I(X_j) \subseteq \mathbb{N}$ for all $j \in [1, \ell]$ to its free variables. An interpretation I that makes φ true, which is denoted by $I \models \varphi$, is called a *model* of φ . The semantics is defined in the usual way. One has

- $I \models \top$ and $I \not\models \perp$ for every I .
- $I \models x_i = x_j$ and $I \models X_i = X_j$ iff (respectively) $I(x_i) = I(x_j)$ and $I(X_i) = I(X_j)$.
- $I \models X_i(x_j)$ iff $I(x_j) \in I(X_i)$.
- $I \models R_i(x_{j_1}, \dots, x_{j_{\alpha_i}})$ iff $(I(x_{j_1}), \dots, I(x_{j_{\alpha_i}})) \in R_i$.
- $I \models \varphi_1 \wedge \varphi_2, I \models \varphi_1 \vee \varphi_2$ and $I \models \neg \varphi$ iff (respectively) $(I \models \varphi_1) \wedge (I \models \varphi_2)$, $(I \models \varphi_1) \vee (I \models \varphi_2)$, and $I \not\models \varphi$.
- $I \models \exists x_i \varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$ iff there exists $n \in \mathbb{N}$ such that $I[x_i = n] \models \varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$.
- $I \models \forall x_i \varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$ iff for every $n \in \mathbb{N}$, one has $I[x_i = n] \models \varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$.

In the two last rules, the notation $I[x_i = n]$, where $n \in \mathbb{N}$, stands for the extension of the interpretation I to one additional first-order variable x_i that takes the value n , i.e., the interpretation such that $I[x_i = n](x_j) = I(x_j)$ for all $j \in [1, k]$ such that $j \neq i$, $I[x_i = n](x_i) = n$, and $I[x_i = n](X_j) = I(X_j)$ for all $j \in [1, \ell]$.

A formula is said to be *satisfiable* if it admits a model.

2.2 Automata

A finite-word or infinite-word automaton is a tuple $\mathcal{A} = (\Sigma, Q, \Delta, Q_0, F)$ where Σ is a finite *alphabet*, Q is a finite set of *states*, $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ is a *transition relation*, $Q_0 \subseteq Q$ is a set of *initial states*, and $F \subseteq Q$ is a set of *accepting states*.

A *path* of $\mathcal{A}$ from q_0 to q_m , with $q_0, q_m \in Q$ and $m \geq 0$, is a finite sequence $\pi = (q_0, a_0, q_1); (q_1, a_1, q_2); \dots; (q_{m-1}, a_{m-1}, q_m)$ of transitions from Δ . The finite word $w \in \Sigma^*$ *read* by π is $w = a_0 a_1 \dots a_{m-1}$; the existence of such a path is denoted by $q_0 \xrightarrow{w} q_m$. A *cycle* is a non-empty path from a state to itself. If $\mathcal{A}$ is a finite-word automaton, then a path π from q_0 to q_m is *accepting* if $q_m \in F$. A word $w \in \Sigma^*$ is *accepted* from the state q_0 if there exists an accepting path originating from q_0 that reads w .

For an infinite-word automaton $\mathcal{A}$, a *run* from a state $q_0 \in Q$ is an infinite sequence $\sigma = (q_0, a_0, q_1); (q_1, a_1, q_2); \dots$ of transitions from Δ . This run reads the infinite word $w = a_0 a_1 \dots \in \Sigma^\omega$. Several types of infinite-word automata have been defined, which differ in their *acceptance condition*. In this paper, we consider the *Büchi* acceptance condition. In a Büchi automaton, the run σ is *accepting* if the set $\text{inf}(\sigma)$ formed by the states q_i that occur infinitely many times in σ is such that $\text{inf}(\sigma) \cap F \neq \emptyset$, i.e., there exists a state in F that is visited infinitely often by σ . A word $w \in \Sigma^\omega$ is *accepted* from the state $q_0 \in Q$ if there exists an accepting run from q_0 that reads w .

For both finite-word and infinite-word automata, a word w is *accepted* by $\mathcal{A}$ if it is accepted from an initial state $q_0 \in Q_0$. The set of all words accepted from a state $q \in Q$ (resp. by $\mathcal{A}$) forms the *language* $L(\mathcal{A}, q)$ accepted from q (resp. $L(\mathcal{A})$ accepted by $\mathcal{A}$). An automaton accepting $L(\mathcal{A}, q)$ can be derived from $\mathcal{A}$ by setting Q_0 equal to $\{q\}$. The language of finite words w read by paths from q_1 to q_2 , with $q_1, q_2 \in Q$, 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 Q_0 equal to $\{q_1\}$ and F equal to $\{q_2\}$. A language is said to be *regular* (resp. *ω -regular*) if it can be accepted by a finite-word (resp. an infinite-word) automaton.

3 Deciding Satisfiability

3.1 Encoding Interpretations

In order to decide whether a formula $\varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$ is satisfiable, Büchi introduced the idea of building an automaton that accepts the set of all models of φ , encoded in a suitable way, and then checking whether it accepts a non-empty language [5, 6].

A simple encoding scheme consists in representing the value of first-order variables x_i in *unary notation*: a number $n \in \mathbb{N}$ is encoded by the infinite word $0^n 1 0^\omega$ over the alphabet $\{0, 1\}$, i.e., by a word in which the symbol 1 occurs only once, at the position given by n . This leads to a compatible encoding scheme for the values of second-order variables X_j : a predicate $P \subseteq \mathbb{N}$ is encoded by

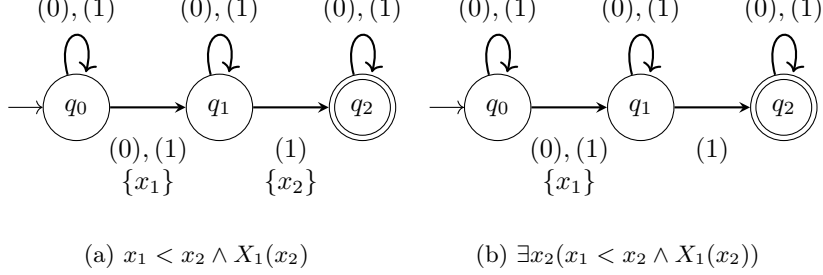


Figure 1: Automata recognizing sets of models.

the infinite word $a_0a_1a_2\dots$ over the alphabet $\{0, 1\}$ such that for every $n \in \mathbb{N}$, a_n satisfies $a_n = 1$ iff $n \in P$, i.e., if $P(n)$ holds.

Encodings for the values of first-order variables $x_1, \dots, x_k$ and second-order variables $X_1, \dots, X_\ell$ can be combined into a single word over the alphabet $\Sigma = \{0, 1\}^{k+\ell}$: a word $w \in \Sigma^\omega$ encodes an interpretation I for those variables iff $w = (a_{0,1}, \dots, a_{0,k+\ell})(a_{1,1}, \dots, a_{1,k+\ell})\dots$, where for each $i \in [1, k]$, $a_{0,i}a_{1,i}\dots$ encodes $I(x_i)$, and for each $j \in [1, \ell]$, $a_{0,k+j}a_{1,k+j}\dots$ encodes $I(X_j)$. Note that not all infinite words over Σ form valid encodings: for each first-order variable x_i , an encoding must contain exactly one occurrence of the symbol 1 for the i -th component of its tuple symbols. Assuming that the set of variables is clear from the context, we write $e(I)$ for the word encoding I with respect to those variables.

3.2 Automata Recognizing Sets of Models

Let S be a set of interpretations for k first-order and ℓ second-order variables. The set of encodings of the elements of S forms a language L over the alphabet $\{0, 1\}^{k+\ell}$. If this language is ω -regular, then we say that an automaton $\mathcal{A}$ that accepts L *recognizes*, or *represents*, the set S . Such an automaton can be viewed as having $k+\ell$ input tapes reading symbols from $\{0, 1\}$, each of these tapes being associated to a variable. Equivalently, we can write the label of a transition $(q_1, (a_1, \dots, a_{k+\ell}), q_2) \in \Delta$ as $\bigvee^{(a_{k+1}, \dots, a_{k+\ell})} V$ where V is the set of the variables x_i , with $i \in [1, k]$, for which $a_i = 1$. In other words, each transition label distinct from ε specifies the set of first-order variables whose value corresponds to this transition, and provides one symbol for each second-order variable. For each $x_i \in V$, we then say that x_i is *associated* to the transition. Note that every transition for which $V \neq \emptyset$ can only be followed at most once in an accepting run. Any automaton recognizing a set of valid encodings can therefore easily be transformed into one in which such transitions do not appear in cycles, and that accepts the same language.

An example of an automaton recognizing the set of models of the formula $\varphi(x_1, x_2, X_1) = x_1 < x_2 \wedge X_1(x_2)$ is given in Figure 1a. For the sake of

clarity, labels of transitions sharing the same origin and destination are grouped together, and empty sets of variables are omitted.

3.3 Decision Procedure

For the automata-based approach to be applicable, it must be possible to construct elementary automata recognizing the models of atomic formulas. This is clearly the case for atoms of the form $x_i = x_j$, $X_i = X_j$ and $X_i(x_j)$, and this property must also hold for each relation R_i that belongs to the structure of interest; in other words, the atomic formula $R_i(x_1, x_2, \dots, x_{\alpha_i})$ must admit a set of models whose encoding is ω -regular. With the encoding of natural numbers introduced in Section 3.1, this is the case in particular for the order relation $x_i < x_j$ and the successor relation $x_j = x_i + 1$. Note that one can easily add supplementary variables to an automaton, by inserting new components in the tuples of its alphabet, and making these components read a symbol 1 at any single position of a run for first-order variables, and any symbol at any position for second-order ones. Reordering the variables is a similarly immediate operation.

After automata recognizing the models of atomic formulas have been obtained, the next step consists in combining them recursively by following the syntactic structure of the formula to be decided. Let us denote by L_φ the language of encodings of all the models of a formula φ , i.e., $L_\varphi = \{e(I) \mid I \models \varphi\}$.

For the Boolean operator $\wedge$, we have $L_{\varphi_1 \wedge \varphi_2} = L_{\varphi_1} \cap L_{\varphi_2}$, where φ_1 and φ_2 are formulas over the same free variables. Similarly, we have $L_{\varphi_1 \vee \varphi_2} = L_{\varphi_1} \cup L_{\varphi_2}$. The case of the complement operator $\neg$ is slightly more complicated, since the complement of a language of encodings systematically contains words that do not validly encode an interpretation. The set of models of a formula $\neg\varphi$ is encoded by the language $\overline{L_\varphi} \cap L_{\text{valid}}$, where L_{valid} is the language of all valid encodings consistent with the free variables of φ . It is easily seen that this language is ω -regular.

It remains to compute the effect of quantifiers. The language $L_{\exists x_i \varphi}$ can be derived from L_φ by removing the i -th component from each tuple symbol, i.e., by applying a mapping $\Pi_{\neq i} : \Sigma^{k+\ell} \rightarrow \Sigma^{k+\ell-1} : (a_1, \dots, a_{k+\ell}) \mapsto (a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_{k+\ell})$ to each symbol of the alphabet. Indeed, the models of $\exists x_i \varphi$ correspond exactly to the models of φ in which the variable x_i is removed. In the rest of this paper, we will use the notation $\Pi_{\neq i}(w)$, where w is a finite or infinite word, to express the result of applying $\Pi_{\neq i}$ to each symbol in w . If L is a language, then we write $\Pi_{\neq i}(L)$ for the language $\{\Pi_{\neq i}(w) \mid w \in L\}$.

Finally, universal quantification can be reduced to existential quantification: for computing $L_{\forall x_i \varphi}$, we use the equivalence $\forall x_i \varphi \equiv \neg \exists x_i \neg \varphi$ which yields $L_{\forall x_i \varphi} = \overline{L_{\exists x_i \neg \varphi}} \cap L_{\text{valid}}$.

3.4 Operations over Automata

We now discuss how the operations over languages mentioned in Section 3.3 can be computed over infinite-word automata. Given automata $\mathcal{A}_1$ and $\mathcal{A}_2$,

automata $\mathcal{A}_1 \cap \mathcal{A}_2$ and $\mathcal{A}_1 \cup \mathcal{A}_2$ accepting respectively $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ and $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ can be obtained by the so-called product construction. The idea consists in building an automaton $\mathcal{A}$ that simulates the combined behavior of $\mathcal{A}_1$ and $\mathcal{A}_2$ on identical input words. The states of $\mathcal{A}$ need to store additional information about the accepting states that are visited in $\mathcal{A}_1$ and $\mathcal{A}_2$. For $\mathcal{A}_1 \cap \mathcal{A}_2$, one ensures that each accepting run of $\mathcal{A}$ correspond to an accepting run in both $\mathcal{A}_1$ and $\mathcal{A}_2$. For $\mathcal{A}_1 \cup \mathcal{A}_2$, the condition is that the run should be accepting in $\mathcal{A}_1$ or $\mathcal{A}_2$, or both. A complete description of the product construction for Büchi automata is given in [17].

Modifying the alphabet of an automaton in order to implement the effect of an existential quantification is a simple operation. As an example, Figure 1b shows an automaton recognizing the set of models of $\exists x_2(x_1 < x_2 \wedge X_1(x_2))$, obtained by removing all occurrences of the variable x_2 from transition labels. Testing whether the language accepted by an automaton is not empty amounts to checking the existence of a reachable cycle that visits at least one accepting state, which is simple as well.

The only problematic operation is complementation, which consists in computing from an automaton $\mathcal{A}$ an automaton that accepts the language $\overline{L(\mathcal{A})}$. Although it preserves ω -regularity, this operation is difficult to perform on Büchi automata [14, 18]. In the context of our decision procedure, it is only useful for applying universal quantifiers. Indeed, other instances of the negation operator in formulas can be pushed inwards until they are applied to atomic formulas, and it is easy to construct the complement of the elementary automata recognizing the models of those atomic formulas, provided that for each relation R_i in the structure of interest, an automaton recognizing $\{(x_1, \dots, x_{\alpha_i}) \in \mathbb{N}^{\alpha_i} \mid \neg R_i(x_1, \dots, x_{\alpha_i})\}$ is available. In order to eliminate the need for complementation, we develop in the next section a direct algorithm for computing the effect of universal quantifiers on automata recognizing sets of models.

4 Universal Quantification

4.1 Principles

Let $\varphi(x_1, \dots, x_k, X_1, \dots, X_\ell)$, with $k > 0$ and $\ell \geq 0$, be a formula. Our goal is to compute an automaton $\mathcal{A}'$ accepting $L_{\forall x_i \varphi}$, given an automaton $\mathcal{A}$ accepting L_φ and $i \in [1, k]$.

By definition of universal quantification, we have $I \models \forall x_i \varphi$ iff $I[x_i = n] \models \varphi$ holds for every $n \in \mathbb{N}$. In other words, $L_{\forall x_i \varphi}$ contains $e(I)$ iff L_φ contains $e(I[x_i = n])$ for every $n \in \mathbb{N}$. Conceptually, we can then obtain $L_{\forall x_i \varphi}$ by defining for each $n \in \mathbb{N}$ the language $S_n = \{e(I) \mid e(I[x_i = n]) \in L_\varphi\}$, which yields $L_{\forall x_i \varphi} = \bigcap_{n \in \mathbb{N}} S_n$.

An automaton $\mathcal{A}'$ accepting $L_{\forall x_i \varphi}$ 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 $\Pi_{\neq i}$ 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.

4.2 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

- $\Sigma' = \Pi_{\neq i}(\Sigma)$.
- $Q' = 2^Q \setminus \{\emptyset\}$.
- 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:

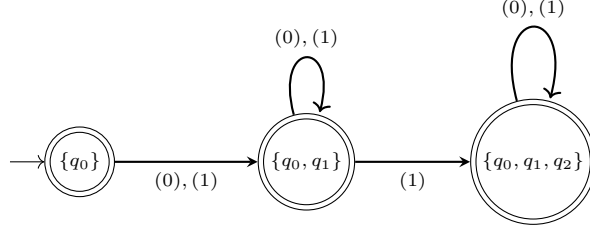


Figure 2: First step of construction for $\forall x_1 \exists x_2 (x_1 < x_2 \wedge X_1(x_2))$.

- * $\bar{q}_1 = \{q_1 \mid (q_1, (a_1, \dots, a_{k+\ell}), q_2) \in T\}$.
- * $\bar{q}_2 = \{q_2 \mid (q_1, (a_1, \dots, a_{k+\ell}), q_2) \in T\}$.
- * 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.

As an example, applying this construction to the automaton in Figure 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 2. For the sake of clarity, unreachable states and states from which the accepted language is empty are not depicted.

4.3 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 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 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 [12, 7] 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 $U_{\bar{q}}^L = L(\mathcal{A}', \bar{q}, \bar{q}) \cap \bigcap_{q \in \bar{q}} U_q$ 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 $u^\omega \in \Pi_{\neq i}(L(\mathcal{A}, q))$, 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 $U_{\bar{q}}^L$ has been obtained, we build a *widget*, in the form of an infinite-word automaton accepting $(U_{\bar{q}}^L)^\omega$, 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.

4.4 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 $u = \Pi_{\neq i}(v)$. This automaton can be constructed in a similar way as one accepting $\Pi_{\neq i}(L(\mathcal{A}, q, q))$ (cf. Sections 2.2 and 3.4), 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 $\Pi_{\neq i}(L(\mathcal{A}, q_1, q_2))$ (cf. Sections 2.2 and 3.4).
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 $L(\mathcal{A}', \bar{q}, \bar{q})$ (cf. Section 2.2).
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 $L_{\forall x_i \varphi}$.

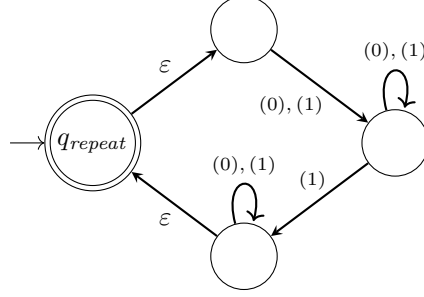


Figure 3: Widget for the state $\{q_0, q_1, q_2\}$.

4.5 Illustration

We illustrate the construction described in Section 4.4 on the automaton $\mathcal{A}'$ in Figure 2, obtained after the first step of the universal quantification procedure applied to the automaton $\mathcal{A}$ in Figure 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 3.
6. The resulting automaton $\mathcal{A}''$ is shown in Figure 4. For the sake of clarity, the states from which the accepted language is empty have been removed.

4.6 Proof of Correctness

Let us show that the automaton $\mathcal{A}''$ constructed according to the procedure developed in Section 4.4 simulates exactly the infinite intersection of the automata $\mathcal{A}_n$, for all $n \geq 0$, as defined in Section 4.2. 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. Note that, since we know that applying a first-order universal quantifier to an automaton recognizing a set of models preserves ω -regularity (cf. Section 3.4), and two ω -regular languages are

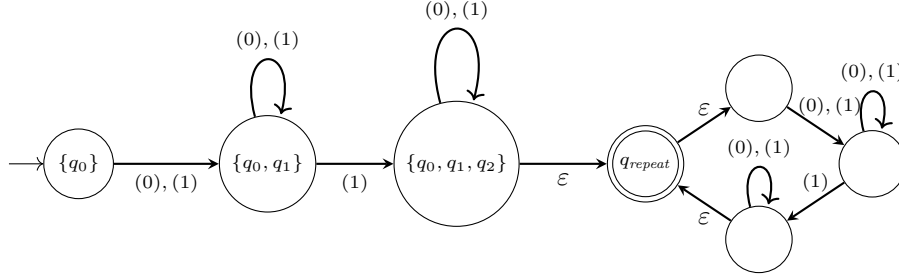


Figure 4: Automaton recognizing the set of models of $\forall x_1 \exists x_2 (x_1 < x_2 \wedge X_1(x_2))$.

equal iff they share the same ultimately periodic words [12, 7], it is sufficient to consider ultimately periodic words. We will rely on the following lemma.

Lemma 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 $q_0 \xrightarrow{u_1} q \xrightarrow{u_2} q \xrightarrow{u_3} q \xrightarrow{u_3} \dots$, 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 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 $\Pi_{\neq i}(w) = w''$, 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 1, there exists a run ρ'' of $\mathcal{A}''$ accepting w'' that is of the form

$$\rho'' : \bar{q}_0 \xrightarrow{u_1} \bar{q} \xrightarrow{\epsilon} q_{repeat} \xrightarrow{u_2} q_{repeat} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} q_{repeat} \xrightarrow{u_3} \dots,$$

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 in Section 4.3.) 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 4.2), 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, $q_j \in \bar{q}$, $\Pi_{\neq i}(u) = u_1 \cdot u_2 \cdot (u_3)^p$, 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 $\Pi_{\neq i}(v) = (u_3)^\omega$. 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 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 $\Pi_{\neq i}(w_n) = w''$, 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

$$\rho_n : q_{n,0} \xrightarrow{a_0} q_{n,1} \xrightarrow{a_1} q_{n,2} \xrightarrow{a_2} \dots q_{n,n} \xrightarrow{\{x_i\}^{a_n}} q_{n,n+1} \xrightarrow{a_{n+1}} \dots,$$

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 1, we can assume w.l.o.g. that ρ_n is of the form

$$\rho_n : q_{n,0} \xrightarrow{a_0} \dots q_{n,n} \xrightarrow{\{x_i\}^{a_n}} q_{n,n+1} \xrightarrow{a_{n+1}} \dots q_{n,j_n} \xrightarrow{u} q_{n,j_n+\delta_n} \xrightarrow{u} q_{n,j_n+2\delta_n} \xrightarrow{u} \dots,$$

where $j_n \geq n + 1$, $\delta_n = |u| > 0$, and $q_{n,j_n} = q_{n,j_n+\delta_n} = q_{n,j_n+2\delta_n} = \dots$ is an accepting state.

By construction of the automaton $\mathcal{A}'$ (cf. Section 4.2), 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 1, $\mathcal{A}'$ admits a run ρ' of the form

$$\rho' : \bar{q}_0 \xrightarrow{u_1} \bar{q} \xrightarrow{u_2} \bar{q} \xrightarrow{u_2} \bar{q} \xrightarrow{u_2} \dots,$$

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 $|u_1| = j_0 + \ell\delta_0$ 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 $\Pi_{\neq i}(w_n) = w''$. 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

$$q_j \xrightarrow{v_j} q_k \xrightarrow{t_j} q_k \xrightarrow{t_j} q_k \xrightarrow{t_j} \dots,$$

with $\Pi_{\neq i}(v_j) = (u_2)^{m_j}$ and $\Pi_{\neq i}(t_j) = (u_2)^{p_j}$ 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 $\Pi_{\neq i}(v) = \Pi_{\neq i}(t) = (u_2)^p$,
- 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

$$\bar{q}_0 \xrightarrow{u_1} \bar{q} \xrightarrow{\varepsilon} q_{\text{repeat}} \xrightarrow{(u_2)^p} q_{\text{repeat}} \xrightarrow{(u_2)^p} q_{\text{repeat}} \xrightarrow{(u_2)^p} \dots$$

that accepts w'' . □

4.7 Complexity

The algorithm introduced in Section 4.4 for computing an automaton accepting $L_{\forall x \varphi}$ from an automaton $\mathcal{A}$ accepting L_φ runs in time $2^{O(|\mathcal{A}|)}$, where $|\mathcal{A}|$ is the size of $\mathcal{A}$. This complexity is tight according to the following result.

Theorem 3. *There exist a family of formulas φ_n for $n \in \mathbb{N}$, and automata $\mathcal{A}^{(n)}$ accepting the corresponding L_{φ_n} , such that $|\mathcal{A}^{(n)}| = O(n)$, and the number of states of any automaton accepting $L_{\forall x \varphi_n}$ is at least $O(2^n)$.*

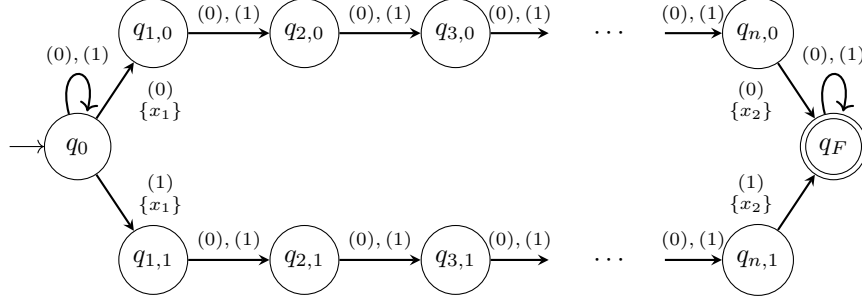


Figure 5: Automaton for $x_2 = x_1 + n \wedge X_1(x_1) \Leftrightarrow X_1(x_2)$.

Proof. The automaton given in Figure 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 parameter.

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 . $\square$

5 Conclusions

This paper introduces a method for directly computing the effect of a first-order universal quantifier on an infinite-word automaton recognizing the set of models of a formula. It is applicable when the first-order variables range over the natural numbers and their values are encoded in unary notation. Among its potential applications, it provides a solution for deciding the first-order theory $\langle \mathbb{N}, < \rangle$ extended with uninterpreted unary predicates.

The operation on regular languages that corresponds to the effect of a universal first-order quantifier has already been studied at the theoretical level [13]. Our contribution is to provide a practical algorithm for computing it, that does not require to complement infinite-word automata. This algorithm has an exponential worst-case time complexity, which is unavoidable since there exist automata for which universal quantification incurs an exponential blowup in their number of states. The main advantage over the complementation-based approach is however that this exponential cost is not systematic, since only a fraction of the possible subsets of states typically need to be constructed. The situation is similar to the subset construction algorithm for determinizing finite-

word automata, which is able to handle in practice automata with millions of states, in spite of its worst-case exponential cost.

Our solution is open to many possible improvements, one of them being to extend the algorithm so as to quantify over several first-order variables in a single operation. For future work, we plan to generalize this algorithm to automata over more expressive structures, such as the automata over linear orderings defined in [4]. This would make it possible to obtain an implementable decision procedure for, e.g., the first-order theory $\langle \mathbb{R}, < \rangle$ with uninterpreted unary predicates [11]. Another challenge would be to develop a similar construction for second-order universal quantification.

Acknowledgments

A preliminary version of this paper appears in [1]. We are grateful to the reviewers of this preliminary version for their comments. This research was partially supported in part by an Amazon Research Award, Fall 2022 CFP. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of Amazon. This work was also partially supported by the Fonds de la Recherche Scientifique - FNRS WEAVE under Grant number T.W028.23.

References

- [1] Bernard Boigelot, Pascal Fontaine, and Baptiste Vergain. Universal first-order quantification over automata. In *Proc. 27th CIAA*, volume 14151 of *LNCS*, pages 91–102. Springer, 2023.
- [2] Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Tr. Comp. Logic*, 6(3):614–633, 2005.
- [3] Bernard Boigelot and Louis Latour. Counting the solutions of Presburger equations without enumerating them. *Theo. Comp. Sc.*, 313(1):17–29, 2004.
- [4] Véronique Bruyère and Olivier Carton. Automata on linear orderings. *Journal of Computer and System Sciences*, 74(1):1–24, 2007.
- [5] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1–6):66–92, 1960.
- [6] J. Richard Büchi. On a decision method in restricted second order arithmetic. In *Proc. Intl. Congr. on Logic, Methodology and Philosophy of Science*, pages 1–12, 1962.
- [7] 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.

- [8] Peter J. Downey. Undecidability of Presburger arithmetic with a single monadic predicate letter. Technical report, Harvard University, 1972.
- [9] Joseph Y. Halpern. Presburger arithmetic with unary predicates is Π_1^1 complete. *The Journal of Symbolic Logic*, 56(2):637–642, 1991.
- [10] Nils Klarlund. Mona & Fido: The logic-automaton connection in practice. In *Proc. 11th CSL Workshop*, volume 1414 of *LNCS*, pages 311–326. Springer, 1997.
- [11] Hans Läuchli and Jon Leonard. On the elementary theory of linear order. *Fundamenta Mathematicae*, 59(1):109–116, 1966.
- [12] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):512–530, 1966.
- [13] Alexander Okhotin. The dual of concatenation. *Theo. Comp. Sc.*, 345(2–3):425–447, 2005.
- [14] Shmuel Safra. On the complexity of omega-automata. In *Proc. 29th FOCS*, pages 319–327. IEEE Computer Society, 1988.
- [15] Thomas R. Shiple, James H. Kukula, and Rajeev K. Ranjan. A comparison of Presburger engines for EFSM reachability. In *Proc. 10th CAV*, volume 1427 of *LNCS*, pages 280–292, 1998.
- [16] Stanislaw O. Speranski. A note on definability in fragments of arithmetic with free unary predicates. *Archive for Mathematical Logic*, 52(5-6):507–516, 2013.
- [17] Wolfgang Thomas. Automata on infinite objects. In Jan Van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B*, pages 133–191. Elsevier and MIT Press, 1990.
- [18] Moshe Vardi. The Büchi complementation saga. In *Proc. 24th STACS*, volume 4393 of *LNCS*, pages 12–22. Springer, 2007.