The Complexity of Sum Automata Expressions with Lipschitz Robustness Application

Nicolas Mazzocchi

Supervisé par Emmanuel Filiot et Jean-François Raskin
Dans l'équipe Formal Methods and Verificiation à l'Université Libre de Bruxelles

21 août 2016

According to the presentation regulations of the report, the first two pages are dedicated to the synthesis of the internship integrity in the research world with a very specific organization

Le contexte général

Ce stage traite l'analyse quantitative de formalismes permettant de définir des fonctions d'un mot vers une valeur. L'étude s'articule autour de deux axes principaux. D'une part, on considère des expressions dont les variables sont évaluées par des weighted automata[1]. D'autre part, on s'intéresse à la décision la lipschitz robustesse i.e. pour deux mots d'input relativement proches, on souhaiterait vérifier si leurs images par une fonction sont linéairement proches. La connexion entre ces axes, réside dans le fait que le modèle des expressions soit un moyen de rendre le problème de lipschitz robustesse décidable. Les questions de robustesse sont depuis longtemps un sujet très actif dans la communauté de vérification, on citera par exemple [32, 20, 21, 10] qui considère principalement la lipschitz robustesse dans le cadres des transductions et des systèmes temporisés.

Le problème étudié

Les modèles que nous étudions sont décomposés en un *valuateur* et un *combinateur*. Le valuateur, a pour but d'évaluer les variables qui dépendent de l'environnement (le mot d'input) et donc potentiellement bruitées. Le combinateur reçoit les variables évaluées par le valuateur et les combines afin de retourner une valeur finale unique. Cette décomposition est en fait une généralisation de la notion d'expression à partir de variables évaluées par des weighted automata déterministes, qui a été introduite dans [6] pour les infinis.

Les problèmes que nous étudions sont la vacuité, l'universalité, la comparaison (une variante de l'inclusion quantitative) et la lipschitz robustesse de nos modèles. Les précédents résultats analysent les problèmes de décisions quantitatif classique (vacuité, l'universalité, inclusion, équivalence) pour les Mean-Payoff Expressions et démontre leur décidabilités [6, 36]. Le sujet consiste à définir des variantes aux Mean-Payoff Expressions dans le cadre de mots finis en considèrent différent valuateurs et combinateurs. Par ailleurs, on souhaiterais décrire une preuve de décidabilité directe (comparable à la construction de Thompson pour les expressions régulière [35]). Et enfin, il s'agit de présenter une analyse fine des complexités, en fixant des paramètres ou bien en codant les valeurs en unaire.

La contribution proposée

L'issue de ce stage propose des modèles dont tous les problèmes de décision quantitatif classique sont dans PSpace. Les valuateurs ont été étendu au weighted automata fonctionnel (contraiement au weighted automata déterministe de [6, 36]), rendant l'expressivité des modèles comparable au weighted automata k-valued (strictement plus expressif). Nous présentons aussi deux caractérisations générales permettant d'obtenir la décidabilité de la lipschitz robustesse et celle des axiomes de distance. Les résultats de complexité sont optimaux et considère les cas suivant.

[1] En français *automate à multiplicité* mais je préfère garder le terme utilisé dans le rapport.

- Valuateur réalisé soit par un unique automate labélisé par des vecteurs soit comme un produit d'automates.
- Combinateur exprimé par une expression ou par une formule existentielle de Presburger.
- Encodage des multiplicités (et du threshold) en unaire ou en binaire.
- Généralisation du problème d'inclusion quantitatif au problème de comparaison afin de ne pas dépendre de l'inclusion des domaines (étant PSpace-Complete).

En ce qui concerne l'historique de la démarche, j'ai tout d'abord donné une preuve que le problème de lipschitz robustesse est en général indécidable pour des fonctions réalisées par des weighted automata. De plus, j'ai montré qu'il est décidable de savoir lorsque nos fonction décrivent une distance, en généralisant à une classe défini par des axiomes plus proche de la théorie des automates. Ceci justifie donc l'utilisation des expressions dans la résolution des problèmes de lipschitz robustesse. Ensuite, mes encadrants m'ont fait découvrir le modèle des reversal bounded counter machines[2] qui permet de présenter une preuve directe que l'inclusion quantitative de nos modèles sont dans PSpace. Puis, les première preuve de NP-Hardness on vu le jour, donnant une motivation à l'extension des combinateurs exprimé par une expression, à une formule existentielle de Presburger afin obtenir la NP-completeness. Finalement, l'étude des complexités a été raffinée au cas d'un encodage unaire.

Les arguments en faveur de sa validité

Les complexités démontrées dans le rapport sont en accord avec les résultats existant [36] mais dans le cadre des mots finis. En outre, la totalité des résultats repose sur d'anciens théorèmes éprouvés. Le Lemme d'Oscar Ibarra [17] pour les counter machines date de 1981. Le théorème de Bruno Sarpellini [33] pour la satisfaisabilité des formules existentielle de Presburger date de 1984. Et de même pour tous les problèmes réduit, *Boolean Emptiness for intersection of regular languages problem* [25] 1977, *Set Partition* [23] 1982, *3-Partition* [16] 1975 et *NFA membership* [22] 1975.

Le bilan et les perspectives

Les résultats de complexité sont succinctement présentés dans la table en Annexe C (dernière page). Notre caractérisation de la décidabilité de la lipschitz robustesse s'applique à toutes les fonctions de la table.

Une analyse similaire pourrait être entreprise dans le cas où les valuateurs sont réalisés par des discounted sum automata ou des ratio automata. Nous pourrions étudier la décidabilité de jeux dont la condition d'acceptation serait exprimé dans l'arithmétique de Presburger. On pourrait aussi envisager que les combinateurs ai une expressivité entendu à quelque chose de plus proche d'un langage de programmation en incluant par exemple les branchements **if**, **then**, **else**. Du coté de la robustesse beaucoup de travaux existent déjà mais ne sont pas couplés avec ce modèle. On pourrais s'intéresser à la synthèse de systèmes robustes à partir d'une spécification logique.

Un mot sur le stage

Durant le premier mois de mon stage j'ai travaillé sur l'adaptation du GAP algorithme du papier [11] de Thomas Colcombet et Laure Daviaud pour des fonctions réalisées par des mean-payoff weighted automata sur des mots infinis réguliés. Cette étude m'a conduit vers des résultats existants sur les jeux d'énergie à informations imparfaites en particulier, je pouvais utiliser certain théorèmes du papier [9]. Néanmoins, la question étant plus difficile qu'elle n'y apparaissait, mes encadrants m'ont alors redirigé vers le domaine de la robustesse. Notez bien que les questions posé sont les même (principalement l'inclusion quantitative) seul change la classe des weighted automata.

Ce stage de recherche, en partie financé par le programme de mobilité ERAMUS, est réalisé à titre obligatoire dans le cadre du Master Parisien de Recherche en Informatique (MPRI). Mon travail sera évalué par Pierre Senellart, Hubert Comon-Lundh et Sophie Laplante (présidente-rapportrice) au cours d'une soutenance. La préférence d'une pratique de langage étranger au lors d'une mobilité Européenne, en vue d'une publication future et surtout la demande de mes encadrants sont autant de raisons qui justifie le choix d'une rédaction en anglais. Je souhaite d'ailleurs leur adresser mes plus chaleureux remerciement pour l'aide, le temps qu'il m'ont consacré ainsi que leur pédagogie. Cher lecteur, bonne lecture.

[2] En français *machines à compteur avec inversement de tendance borné* mais je préfère garder le terme utilisé dans le rapport.

The Complexity of Sum Automata Expressions with Lipschitz Robustness Application

Mazzocchi Nicolas [3]
Master thesis under the direction of
Emmanuel Filiot [4] & Jean-François Raskin [5]

August 21, 2016

Abstract

We propose to study a computational model inspired by Mean-Payoff Expressions [6], which separates the valuations of the environmental variables (potentially noisy) and the program, also called combiner, that compute the final result. In this report, the inputs will be represented by a finite word. Environmental variables are evaluated by a functional sum weighted automaton and the program is written either as an expression using operations, min, max, plus, minus, or as an existential Presburger formula. We shown that all classical quantitative decision problems are in PSpace and we refine many complexity cases. The valuations can be given by a single automaton with vectors or as a product, encoding of weights can be binary or unary. Finally, we study the decidability of lipschitz robustness for such model, and we present a general characterization for it decidability.

Contents

[3]MPRI student at the École Normale Supérieur de Cachan, Université Paris-Saclay – France
[4]FNRS Research associate at Université Libre de Bruxelles – Belgium
[5]Professor of Computer Science at Université Libre de Bruxelles – Belgium

1 Introduction

Nowadays, most computational embedded systems for critical applications use a *reactive system*. These implementations maintain a continuous interaction with the environment in which they operate. But, any flaw in such a critical computation can have catastrophic consequences. Yet, they exhibit several features that make them difficult to design correctly, like resources and real-time constraints, concurrency, parallelism and so on. To ensure the design of reactive computer systems that are dependable, safe and efficient, researchers and industry have advocated the use of so-called *formal methods*, that rely on mathematical models to express precisely and analyze the behaviors of those systems.

In 1950, Stephen Cole Kleene in [24], and Michael Oser Rabin & Dana Scott in [29] initiated *Boolean* formal methods elegantly and efficiently supported by *automata theoretic methods*. So, in the traditional approach, a system is either *correct* when it transforms all allowed input streams of the environment into input-output streams accepted by the specification, or *incorrect* otherwise. However, when considering reactive systems, it is often desirable to adopt a more *quantitative* approach in which the performance of the system (and not only its correctness) can be modeled and analyzed.

The idea of *Weighted Automata* (WA for short), is to associate each word to a value from a computation in its semiring, for example the tropical semiring $\langle \mathbb{N} \cup \{\infty\}, \min, +, \infty, 0\rangle$. In addition, all classical problems (emptiness, universality, inclusion, equivalence) can be extended to this framework [7]. WA were introduced in 1961 by Schützenberger in [34], and generalized in [14]. One may refer to books [13] and [31] for a complete overview. Later, several expressiveness variants have been introduced for trees [2], alterning automata [8], timed automata [5], lattice automata [19]. Even Büchi's and Elgot's fundamental theorems were extended with a weighted version of MSO logics in [12]. Nevertheless, in 1994 Krob reduced Hilbert's 10^{th} problem (solving a diophantine equation) to proves the undecidability of the universality problem for the tropical semiring with domain $\mathbb{Z} \cup \{\infty\}$. This implies undecidability of inclusion. An alternative proof by reducing from the halting problem of a 2-counter machine is given in [1].

Thus, *Quantitative Expressions* have been defined to dodge undecidability while keeping a powerful expressiveness. The computation of expression setting is separated in a valuation of variables during input word reading and, a combiner which uses them to construct the final result. The first quantitative expressions considered, *Mean-Payoff Expressions*, were introduced in [6] with valuations realized by a deterministic Mean-Payoff WA and combiner operations min, max, minus, plus. The proof used, to show that the quantitative decision problems are decidable for these expressions, presents a geometric approach and yields a complexity 4ExpTime. The result has been improved in [36] with a new proof based on cycle analysis in the automata, the problem is finally PSpace-Complete.

In the report we consider valuations performed by functional Sum-WA in $\mathbb{Z}$ over finite words. The importance of handling functional automata rather than deterministic ones is because their expressiveness is comparable to k-valued weighted automata (a k-valued weighted automaton can be decomposed in to a k unions of functional weighted automata, unions of deterministic ones are strictly less expressive [39]). Furthermore, we solve slightly different problems : Classical inclusion problem (respectively equivalence) check the inclusion of the domain (respectively equivalence) which is PSpace-Complete [27]. But this hardness is not satisfactory in the sense that this is not the heart of the problem. So, in the next section, we define *comparison* problems which does not depend on this inclusion (respectively equivalence). Section 3 shows a simpler proof of the complexity result stated above and some hardness results. Section 4 extends the expressiveness of the combiner to existential Presburger logic. Finally, Section 5 uses expressions in the *lipschitz robustness* problem defined in [32] and [20]. A function is called K-lipschitz robust if the perturbation in its output is at most K times the perturbation in its input, where the input and output perturbation is defined by a distance. Obviously, in general the question is undecidable for weighted automata but, when the function and distance are performed by our valuation and combiner, the problem becomes decidable. Lastly, we present an application of the lipschitz robustness problem inspired from [4, 3].

2 Preliminaries

In this section, we introduce the main models used in this work, functional Sum-WA and reversal bounded counter machine. WA, labeled in $\mathbb{Z}^k$, will be used to define valuations and reversal bounded counter machine will be the model to which all our valuations and combiners can be translated. We also present a formalization of the problems that we will consider throughout the report.

Definition : Language Theory Notations

An *alphabet* Σ is a non-empty finite set, its elements are called letters, characters or symbols. The empty letter is denoted by ε and the alphabet Σ augmented with ε will be Σ_ε. More generally, the alphabet Σ augmented with a fresh symbol $\#$ will be $\Sigma_\#$. In this document, we note alphabets $\Sigma, \Gamma, \mathcal{V}$ and for letters we use a, b, α, β and σ.

A *word* on an alphabet Σ is a (potentially empty) sequence of symbols belonging to Σ. The empty word is denoted by ε (like empty letter). The length of a word w is the number of symbols that compose it, noted $|w|$ and $|w|_a$ for the occurrences number of some letter $a \in \Sigma$. In particular, $|\varepsilon| = 0$. Given a word w, for each $i \in [1..|w|]$ the symbol at position i in w can be written $w[i]$. In this document, we consider that words are finite, and we note them u, v, w. A set of words on an alphabet is called a *language*. The set of all words over an alphabet Σ is denoted by Σ^* and the set of regular languages over Σ is REG_Σ.

Definition : Weighted Automaton (over $\mathbb{Z}$)

A $\mathbb{Z}^k$-WA over a semiring $(\mathbb{Z}^k, \boxplus, \boxtimes, \mathbb{0}, \mathbb{1})$ is a tuple $\langle \Sigma, Q, s, F, \Delta, \gamma \rangle$ where :

- Σ an alphabet
- Q a finite set of states
- $s \in Q$ an initial state
- $F \subseteq Q$ a set of accepting states
- $\Delta : Q \times \Sigma \times Q$ a transition relation
- $\gamma : \Delta \to \mathbb{Z}^k$ a weight function

The tuple $(p, a, q) \in \Delta$ is a transition from state p to q which read the input letter a with $\gamma(p, a, q)$ the weight vector. Such transition in an automaton $\mathcal{A}$ can be noted, $p \xrightarrow{a \,|\, \gamma(p,a,q)}_{\mathcal{A}} q$. Also, we will talk about the maximal weight ℓ, other words for saying $\ell := \max\{|\gamma(p, a, q)[j]| : (p, a, q) \in \Delta \wedge j \in [1..k]\} \in \mathbb{Z}$.

A *path* ρ over $w := a_1 \dots a_n \in \Sigma^*$ is a sequence $\rho := q_1 a_1 \dots a_n q_{n+1}$ where $(q_j, a_j, q_{j+1}) \in \Delta$ for each $j \in [1..n]$. The *cost* of a path is defined by $\mathbf{Cost}(q_1 a_1 \dots a_n q_{n+1}) := \big(\boxtimes_{j \in [1..n]} \gamma(q_j, a_j, q_{j+1})\big) \boxtimes \mathbb{1}$. A path is *accepting* if it starts in the initial state s and ends in some state of F, we note $\mathbf{Path}(w)$ the set of accepting paths over w.

An *execution* (or *run*) over w adds the cost of all accepting paths over w, $[\![\mathcal{A}]\!](w) := \big(\boxplus_{\rho \in \mathbf{Path}(w)} \mathbf{Cost}(\rho)\big) \boxplus \mathbb{0}$. Thus the semantics of $\mathcal{A}$ a $\mathbb{Z}^k$-WA is the partial function $[\![\mathcal{A}]\!] : \Sigma^* \to \mathbb{Z}^k$. We denote $\mathcal{L}_\mathcal{A} := dom([\![\mathcal{A}]\!])$, the language of $\mathcal{A}$ and the size of $\mathcal{A}$ is defined as $|\mathcal{A}| := |Q| + |\Delta| + k \times \log_2(\ell)$.

The most considered type of WA in the literature are tropical $\mathbb{Z}^k$-WA where $\boxtimes := +$ and $\boxplus := \min$. In the remainder, we mainly use a restriction of tropical automata, that for each word w, all its accepting paths have the same cost and $\boxplus$ is idempotent, named functional $\mathbb{Z}^k$-SumWA. This property can be decided in PTime [15].

Definition : Product of $\mathbb{Z}^k$Sum-WA

Let $\mathcal{A} := \langle \Sigma, Q_\mathcal{A}, s_\mathcal{A}, F_\mathcal{A}, \Delta_\mathcal{A}, \gamma_\mathcal{A} \rangle$ be a $\mathbb{Z}^n$Sum-WA and $\mathcal{B} := \langle \Sigma, Q_\mathcal{B}, s_\mathcal{B}, F_\mathcal{B}, \Delta_\mathcal{B}, \gamma_\mathcal{B} \rangle$ be a $\mathbb{Z}^m$-WA. We define the $\mathbb{Z}^{n+m}$Sum-WA product $\mathcal{A} \times \mathcal{B} := \langle \Sigma, Q_\mathcal{A} \times Q_\mathcal{B}, (s_\mathcal{A}, s_\mathcal{B}), F_\mathcal{A} \times F_\mathcal{B}, \Delta, \gamma \rangle$ where Δ and γ as follows.

$$p_1 \xrightarrow{a \,|\, v_1}_{\mathcal{A}} q_1 \wedge p_2 \xrightarrow{a \,|\, v_2}_{\mathcal{B}} q_2 \Rightarrow (p_1, p_2) \xrightarrow{a \,|\, \binom{v_1}{v_2}}_{\mathcal{A} \times \mathcal{B}} (q_1, q_2)$$

Definition : Counter Machine

A k-counter machine is a tuple $\langle \Sigma, X, Q, s, F, \Delta, \lambda \rangle$ where :

- Σ an alphabet
- X a set of counters interpreted in $\mathbb{N}$ such that $|X| = k$
- Q a finite set of states
- $s \in Q$ an initial state
- $F \subseteq Q$ a set of accepting states
- $\Delta : Q \times \Sigma_\varepsilon \times \{\mathbf{decr}, \mathbf{nop}, \mathbf{incr}\}^k \times Q$ a transition relation
- $\lambda : \Delta \to \{\mathbf{equal}, \mathbf{greater}, \mathbf{top}\}^k$ a zero guards

with

$\mathbf{decr} := x \mapsto x - 1$

$\mathbf{nop} := x \mapsto x$

$\mathbf{incr} := x \mapsto x + 1$

$\mathbf{zero} := x \mapsto x = 0$

$\mathbf{top} := x \mapsto \top$

$\mathbf{greater} := x \mapsto x > 0$

The tuple $(p, \sigma, \tau, q) \in \Delta$ is a transition from state p to q which reads the input letter σ (possibly ε) and applies counter transformations vector τ such that $\sum_{x=1}^{k} (\tau[x] \neq \mathbf{nop}) \leq 1$, i.e τ changes at most one counter. Since counters are valued over $\mathbb{N}$, the function λ must satisfy $\forall x \in [1..k].\ (\tau[x] = \mathbf{decr}) \implies (\lambda(p, \sigma, \tau, q)[i] = \mathbf{greater})$ i.e λ hardcodes the fact that counters are always positive. Such transition in a machine $\mathcal{M}$ can be noted,

$p \xrightarrow{\sigma,\lambda(p,a,\tau,q)\,|\,\tau}_{\mathcal{M}} q$. In figures, we can hide **top** and **nop**, write conjunctions and disjunctions in the zero guards or transform a constant number of counters, see FIGURE 2.1 for further explanations.

A *path* ρ over $w := \sigma_1 \dots \sigma_n \in \Sigma_\varepsilon^*$ is a sequence $\rho := q_1\sigma_1\tau_1 \dots \sigma_n\tau_n q_{n+1}$ where $(q_j, \sigma_j, \tau_j, q_{j+1}) \in \Delta$ for each $j \in [1..n]$. A path is *accepting* if it starts in the initial state s and ends in some state of F, we note $\mathbf{Path}(w)$ the set of accepting paths over w. Counter transformations and zero guards can be extended to a path by induction on its length by the following function which takes a counter valuation ν.

$$\overline{\tau}_\rho := \nu \mapsto \begin{cases} \nu & \textbf{if } \rho = q \\ [\overline{\tau}_{q'\rho'} \circ \tau](\nu) & \textbf{if } \rho = q\sigma\tau q'\rho' \end{cases}$$

If the path has no transition, the counter transformations along path are the identity function and zero guards along path are obviously always satisfied. Otherwise, the counter transformations along a path are the ordered memberwise composition of counter transformations.

The zero guards along a path are the memberwise conjunction between zero guards of the first transition and zero guards along to the remaining path after applying the first counter transformation.

$$\overline{\lambda}_\rho := \nu \mapsto \begin{cases} \top & \textbf{if } \rho = q \\ \lambda(q,\sigma,\tau,q')(\nu) \wedge [\overline{\lambda}_{q'\rho'} \circ \tau](\nu) & \textbf{if } \rho = q\sigma\tau q'\rho' \end{cases}$$

A *run* over w and for an initial counter valuation ν, is an accepting path ρ over w if $\forall x \in [1..k].\ \overline{\lambda}_\rho(\nu)[x] = \textbf{true}$, i.e ν allows to satisfy all zero guards along path. Thus, the semantics of a counter machine $\mathcal{M}$ is the partial function defined by $[\![\mathcal{M}]\!](w,\nu) := \{\overline{\tau}_\rho(\nu) : \rho \in \mathbf{Path}(w) \wedge \forall x \in [1..k].\ \overline{\lambda}_\rho(\nu)[x]\}$. For an initial counter valuation ν, we call the language of $\mathcal{M}$ the set $\mathcal{L}^\nu_\mathcal{M} := \{w : [\![\mathcal{M}]\!](w,\nu) \neq \varnothing\}$ and we note $\mathbb{0}$ the zero counter valuation. We say a machine $\mathcal{M}$ has r *reversals* to describe the fact that each counter of $\mathcal{M}$ changes from a increasing mode to decreasing mode and vice-versa at most r times.

Example

In the figure beside, labels of the one reversal counter machine use lots of macro to make a more understandable figure. The green bubble, each disjunction member should make one edge associated with **top** on the other guards. There are not illegal operations, but the red bubbles labels should be $\langle a, (\mathbf{top}, \mathbf{top}) \,|\, (\mathbf{incr}, \mathbf{nop})\rangle$ & $\langle b, (\mathbf{top}, \mathbf{top}) \,|\, (\mathbf{nop}, \mathbf{incr})\rangle$ for bottom one, and for the top one $\langle \varepsilon, (\mathbf{equal}, \mathbf{equal}) \,|\, (\mathbf{not}, \mathbf{nop})\rangle$. Finally, there are two operations the blue bubble, the corresponding state would be split into two linked with ε-transitions.

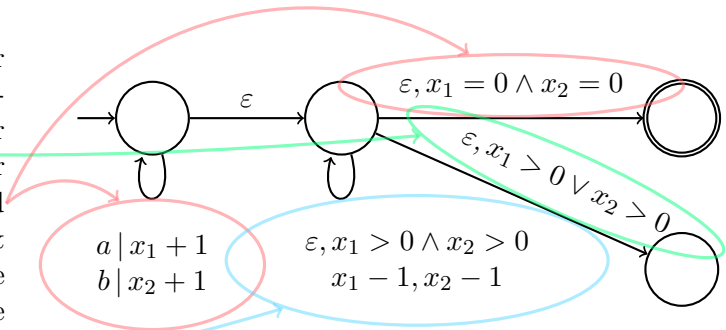


FIGURE 2.1 : $\mathcal{L}^{\mathbb{0}}_\mathcal{M} := \{w : |w|_a = |w|_b\}$

Definition : Counter Machine Combiners

Given a k_1-counter machine $\mathcal{M}_1 := \langle \Sigma, X_1, Q_1, s_1, F_1, \Delta_1\rangle$ with n_1 states, r_1 reversals and a k_2-counter machine $\mathcal{M}_2 := \langle \Sigma, X_2, Q_2, s_2, F_2, \Delta_2\rangle$ with n_2 states, r_2 reversals, we want to combine their computations into a new one according to several operations.

Sequential (concatenation) : Noted $\mathcal{M}_1 \cdot \mathcal{M}_2$, the concatenation is a k-counters machine $\mathcal{M} := \langle \Sigma, X_1 \cup X_2, Q_1 \cup Q_2, s_1, F_2, \Delta\rangle$ with $\max\{k_1, k_2\} \leq k \leq k_1 + k_2$ (according to the number of counter used in both machines), $n := \mathcal{O}(n_1 + n_2)$ states and r reversals such that $\max\{r_1, r_2\} \leq r \leq r_1 + r_2$. The relation Δ is defined by the union of Δ_1, Δ_2 and ε-transitions without effect on counters, from each states of F_1 to s_2. The semantics are as follows.

$$\forall \nu : X \to \mathbb{Z}.\ \mathcal{L}^\nu_\mathcal{M} = \mathcal{L}^\nu_{\mathcal{M}_1} \cdot \mathcal{L}^\nu_{\mathcal{M}_2} \wedge \forall w \in \Sigma^*.\ \exists u, v \in \Sigma^*.\ [\![\mathcal{M}]\!](w,\nu)[x] := \begin{cases} [\![\mathcal{M}_1]\!](u,\nu)[x] & \textbf{if } x \in X_1 \setminus X_2 \\ [\![\mathcal{M}_2]\!](v,\nu)[x] & \textbf{if } x \in X_2 \setminus X_1 \\ [\![\mathcal{M}_2]\!]\,(v, [\![\mathcal{M}_1]\!](u,\nu)) & \textbf{if } x \in X_1 \cap X_2 \end{cases}$$

Non-deterministic (union) : Noted $\mathcal{M}_1 \,||\, \mathcal{M}_2$, the union is a k-counter machine $\mathcal{M} := \langle \Sigma, X_1 \uplus X_2, Q_1 \cup Q_2 \cup \{q\}, q, F_1 \cup F_2, \Delta\rangle$ with $k = k_1 + k_2$ (separate counters)[6] $n := \mathcal{O}(n_1 + n_2)$ states and r reversals such that $r := \max\{r_1, r_2\}$. The relation Δ is defined by the union of Δ_1, Δ_2 and two ε-transitions without effect on counters, from the new initial state q to s_1 and to s_2. The semantics are as follows.

$$\forall \nu : X \to \mathbb{Z}.\ \mathcal{L}^\nu_\mathcal{M} := \mathcal{L}^\nu_{\mathcal{M}_1} \cup \mathcal{L}^\nu_{\mathcal{M}_2} \wedge \forall w \in \Sigma^*.\ [\![\mathcal{M}]\!](w,\nu)[x] := \begin{cases} [\![\mathcal{M}_1]\!](w,\nu)[x] & \textbf{if } x \in X_1 \\ [\![\mathcal{M}_2]\!](w,\nu)[x] & \textbf{if } x \in X_2 \end{cases}$$

[6] Union can be defined without this restriction but we don't need it

Definition : Decision Problems

Given two finitely presented functions $f, g : \Sigma^* \to \mathbb{Z}$, a threshold $c \in \mathbb{Z}$, $\asymp \in \{\leq, <, =, >, \geq\}$ and $\succ \in \{=, >, \geq\}$.

Zero-Emptiness : $\exists u \in dom(f).\ f(u) \asymp 0$

Universality : $\forall u \in dom(f).\ f(u) \asymp c$

Emptiness : $\exists u \in dom(f).\ f(u) \asymp c$

Comparison : $\forall u \in dom(f) \cap dom(g).\ f(u) \succ g(u)$

3 Quantitative Expressions

Initially introduced in [6] and [36], quantitative expressions have been introduced as a terms whose can be constants are deterministic WA. The scission between automata and operations (that we call respectively valuations and combiners) is the result of our generalization. In the following, functional $\mathbb{Z}^k$Sum-WA will be our valuations in two different ways : There can be a single automaton with vector on its transitions as we have defined in preliminaries or, one automaton for each variable (given as a product). Obviously, if we actually compute the product, we obtain the single automaton, but it can result in an exponential blow up. For this reason, our complexity proofs will distinguish these two presentations. Remark, if the number of variables is fixed, the product can be done for free.

In this section, we show an alternative proof to that in [36], of the fact that all decision problems for sum automata expressions over finite word given as a product are in PSpace. The idea is to translate the valuation and the combiner to a reversal bounded counter machine like the Thompson construction for regular expressions [35]. We also present several hardness results. The problem is PSpace-Hard if valuation is given as a product (same straightforward proof than [36]) and NP-Hard otherwise (which separates binary and unary cases).

Definition : Expression

An expression over the set of variables X is a term generated by the following grammar where variables are interpreted in $\mathbb{Z}$.

$$E := 0 \mid 1 \mid x \in X \mid -E \mid \min\{E_1, E_2\} \mid \max\{E_1, E_2\} \mid E_1 + E_2$$

An expression is associated with its semantic tree (described beside for $E := \min\{-x_1, 1 + x_2\}$). We define the language $\mathcal{T}(E) \subseteq \{\alpha, \beta\}^*$ to describe all corrects path in the semantic tree of E. Let's note E_u to describe the sub-expression at position $u \in \mathcal{T}(E)$. The proposition $\mathbf{Var}(E_u)$ holds if E_u is a variable. The semantics of an expression E, is $[\![E]\!] : (X \to \mathbb{Z}) \to \mathbb{Z}$ that takes a valuation and return a value in $\mathbb{Z}$. The size of an expression E, is $|E| := |\mathcal{T}(E)|$.

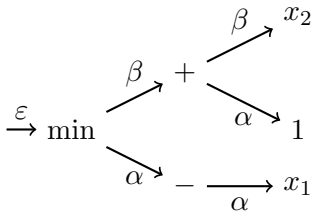


FIGURE 3.1 : Semantic tree

3.1 Translate valuations and expressions to reversal bounded counter machine

Most of the considered decision problems will be reduced to emptiness of reversal bounded counter machines. The interest of such a translation is the use of a very strong complexity result which : given a reversal bounded counter machine, one can compute a threshold for the smallest number of transitions leading to accepting. Then, we show two lemmas to translate respectively functional $\mathbb{Z}^k$Sum-WA and expressions into a reversal bounded counter machine.

Lemma 3.1

Let $\mathcal{A}$ be a functional $\mathbb{Z}^k$Sum-WA with maximum weight ℓ and m transitions. One can construct a $2k$-counter machine $\mathcal{M}$ whose $X := \{x_1^+, x_1^- \dots x_k^+, x_k^-\}$ its counter set (all are increasing only) and with $\mathcal{O}(mk\ell)$ transitions such that :

$$\mathcal{L}_{\mathcal{A}} = \mathcal{L}^0_{\mathcal{M}} \wedge \forall u \in \mathcal{L}_{\mathcal{A}}.\ \forall j \in [1..k].\ [\![\mathcal{A}]\!](u)[j] = [\![\mathcal{M}]\!](u, \mathbb{0})[x_j^+] - [\![\mathcal{M}]\!](u, \mathbb{0})[x_j^-]$$

Proof

Recall that counters of $\mathcal{M}$ are valued in $\mathbb{N}$. Thus, we will encode the j^{th} value of $\mathcal{A}$ by the couple (x_j^+, x_j^-) representative of fictive counter x_j. So x_j^+ represents the increments applied to the fictive counter while x_j^- represents the decrements. Moreover, we wish that $\mathcal{M}$ have the least possible reversals (in particular 0). So even if the counters of $\mathcal{M}$ were interpreted in $\mathbb{Z}$, this separation is necessary.

The construction of $\mathcal{M}$ is easy, for any $p \xrightarrow{a|c_1,\dots,c_k}_{\mathcal{A}} q$ we create the pseudo-transition $p \xrightarrow{a,|\,(c_1+1,\dots,c_k+1)}_{\mathcal{M}} q$ (see technical details in Appendix B). Furthermore, we can checks that the real number of created transition is $\mathcal{O}(km\ell)$ and all are incremental.

Since $\mathcal{A}$ is functional, for any fixed word $u \in \mathcal{L}_{\mathcal{A}}$, the cost of a run over u is exactly the cost of any accepting path over u. And by construction, there is a bijection between the accepting paths in $\mathcal{A}$ and accepting paths in $\mathcal{M}$. Therefore, all runs of $\mathcal{M}$ over u lead to the same counter valuation and then $[\![\mathcal{M}]\!]$ is a function. We conclude by saying that the zero valuation is the neutral valuation for the sum operator. □

Lemma 3.2

Let E be an expression over X. One can construct a $2|E|$-counter machine $\mathcal{M}$ whose $X := \bigcup_{u\in\mathcal{T}(E)}\{x_u^+, x_u^-\}$ its counter set (one reversal, increasing-decreasing) and with $\mathcal{O}(|E|)$ transitions such that :

$$\forall \nu : X \to \mathbb{Z}.\ [\![E]\!](\nu) = [\![\mathcal{M}]\!](\varepsilon, \hat{\nu})[x_\varepsilon^+] - [\![\mathcal{M}]\!](\varepsilon, \hat{\nu})[x_\varepsilon^-]$$

where $\forall u \in \mathcal{T}(E).\ \ \hat{\nu}(x_u^+) := \begin{cases} \nu(E_u) & \text{if } \mathbf{Var}(E_u) \wedge 0 < \nu(E_u) \\ 0 & \text{otherwise} \end{cases}$ $\hat{\nu}(x_u^-) := \begin{cases} -\nu(E_u) & \text{if } \mathbf{Var}(E_u) \wedge \nu(E_u) < 0 \\ 0 & \text{otherwise} \end{cases}$

Proof

Recall that the counters of $\mathcal{M}$ are valued in $\mathbb{N}$ while variables in expressions can be valued by negative numbers. Thus we will encode the result of each operator by the couple (x_u^+, x_u^-) representative of the fictive counter x_u where $u \in \mathcal{T}(E)$. So x_u^+ represent the increments applied to the respective fictive counter while x_u^- represent the decrements. All fictive counters must respect some normal form : one counter of the couple must be zero. We show the construction by structural induction on E. Let us note E_u to describe the sub-expression at position $u \in \mathcal{T}(E)$ in the semantic tree and we call root couple the counter couple associated to the root operation.

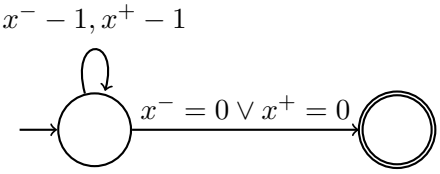


FIGURE 3.2 : $\mathcal{M}_{norm}(x)$

Constants : Case $E_u := C$ where $C \in \{0, 1\}$. Obvious.

Variable : Case $E_u := x$. We simply normalizes the couple (x^-, x^+), see FIGURE 3.2.

Minus : Case $E_u := -E_{u\alpha}$. By induction hypothesis $\mathcal{M}_{u\alpha}$ over $X_{u\alpha} := \bigcup_{u\in\mathcal{T}(E_{u\alpha})}\{x^+, x^-\}$ denotes $E_{u\alpha}$ with the root couple $(x_{u\alpha}^-, x_{u\alpha}^+)$. After the computation of $\mathcal{M}_{u\alpha}$, we proceed to the assignments $x_u^+ := x_{u\alpha}^-$ and $x_u^- := x_{u\alpha}^+$ realized by $\mathcal{M}_{minus}(x_{u\alpha}, x_u)$, see FIGURE 3.3. To do this, we construct $\mathcal{M}_u := \mathcal{M}_{u\alpha} \cdot \mathcal{M}_{minus}(x_{u\alpha}, x_u)$. Note that this transformation preserves counter normalization.

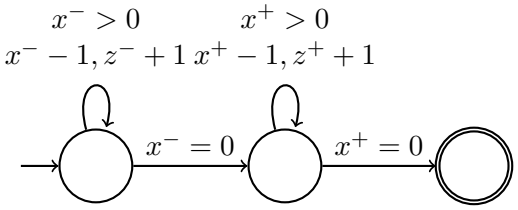


FIGURE 3.3 : $\mathcal{M}_{minus}(x, z) = z := -x$

Sum : Case $E_u := E_{u\alpha} + E_{u\beta}$. By induction hypothesis $\mathcal{M}_{u\alpha}$ over $X_{u\alpha} := \bigcup_{u\in\mathcal{T}(E_{u\alpha})}\{x^+, x^-\}$ and $\mathcal{M}_{u\beta}$ over $X_{u\beta} := \bigcup_{u\in\mathcal{T}(E_{u\beta})}\{x^+, x^-\}$ denotes respectively $E_{u\alpha}$ with the root couple $(x_{u\alpha}^-, x_{u\alpha}^+)$ and $E_{u\beta}$ with the root couple $(x_{u\beta}^-, x_{u\beta}^+)$. After the computation of $\mathcal{M}_{u\alpha}$ and $\mathcal{M}_{u\beta}$, we proceed to the assignments $x_u^+ := x_{u\alpha}^+ + x_{u\beta}^+$ and $x_u^- := x_{u\alpha}^- + x_{u\beta}^-$ realized by $\mathcal{M}_{sum}(x_{u\alpha}, x_{u\beta}, x_u)$, see FIGURE 3.4. To do this, we construct $\mathcal{M}_u := \mathcal{M}_{u\alpha} \cdot \mathcal{M}_{u\beta} \cdot \mathcal{M}_{sum}(x_{u\alpha}, x_{u\beta}, x_u) \cdot \mathcal{M}_{norm}(x_u)$. Note that $\mathcal{M}_{sum}$ does <u>not</u> keep counter normalization, then the norm gadget will be always concatenate after the sum gadget.

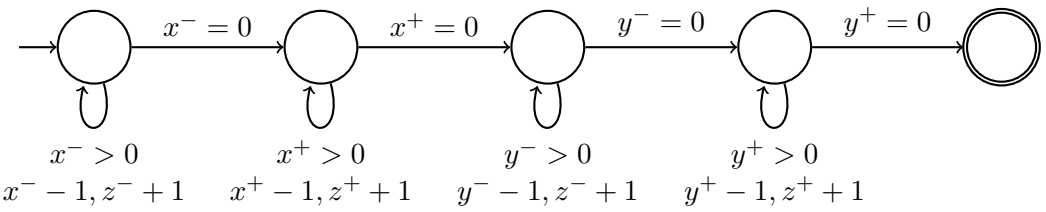


FIGURE 3.4 : $\mathcal{M}_{sum}(x, y, z) = z := x + y$

Minimum / Maximum : Case $E_u := E_{u\alpha} \bowtie E_{u\beta}$ where $\bowtie \in \{\min, \max\}$. By induction hypothesis $\mathcal{M}_{u\alpha}$ over $X_{u\alpha} := \bigcup_{u\in\mathcal{T}(E_{u\alpha})}\{x^+, x^-\}$ and $\mathcal{M}_{u\beta}$ over $X_{u\beta} := \bigcup_{u\in\mathcal{T}(E_{u\beta})}\{x^+, x^-\}$ denotes recpectively $E_{u\alpha}$ with the

root couple $(x^-_{u\alpha}, x^+_{u\alpha})$ and $E_{u\beta}$ with the root couple $(x^-_{u\beta}, x^+_{u\beta})$. After the computation of $\mathcal{M}_{u\alpha}$ and $\mathcal{M}_{u\beta}$ and by counter normalization, for all couple one counter is zero and then we can separate the four sign cases to select the interesting parameter, see FIGURE 3.5 and FIGURE 3.6. To do this, we construct $\mathcal{M}_u := \mathcal{M}_{u\alpha} \cdot \mathcal{M}_{u\beta} \cdot \mathcal{M}_{\bowtie}(x_{u\alpha}, x_{u\beta}, x_u)$. Note that this transformation preserves counter normalization.

In FIGURE 3.5, one can take branching east or west only if both fictive counters x and y have an opposite sign. In this case, we just copy the negative one in z. The machine can take the south branch, only if both fictive counters x and y are positives. In this case, we decrease both simultaneously and increase z until one of them becomes zero. In the last case, we increase both simultaneously and decrease z until one of them becomes zero, then we continue in the first case. The FIGURE 3.6 work with a symmetric idea.

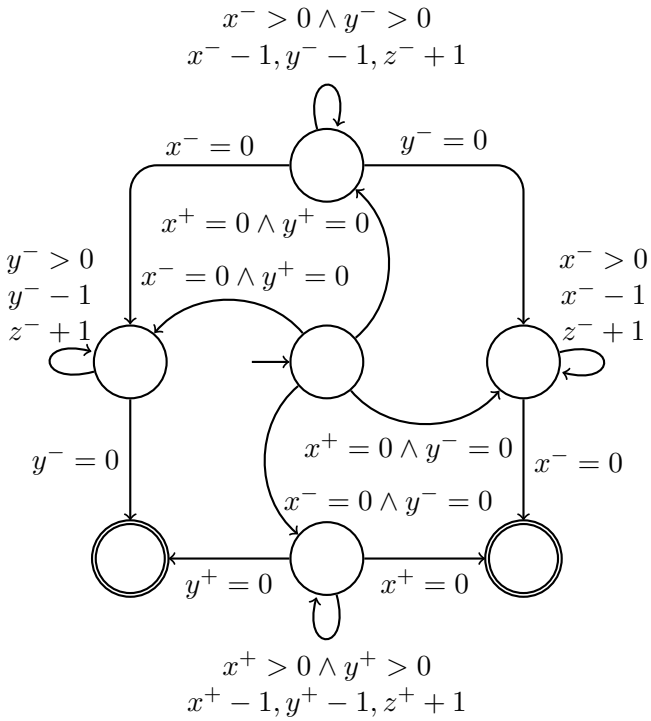


FIGURE 3.5 : $\mathcal{M}_{min}(x, y, z) = z := \min\{x, y\}$

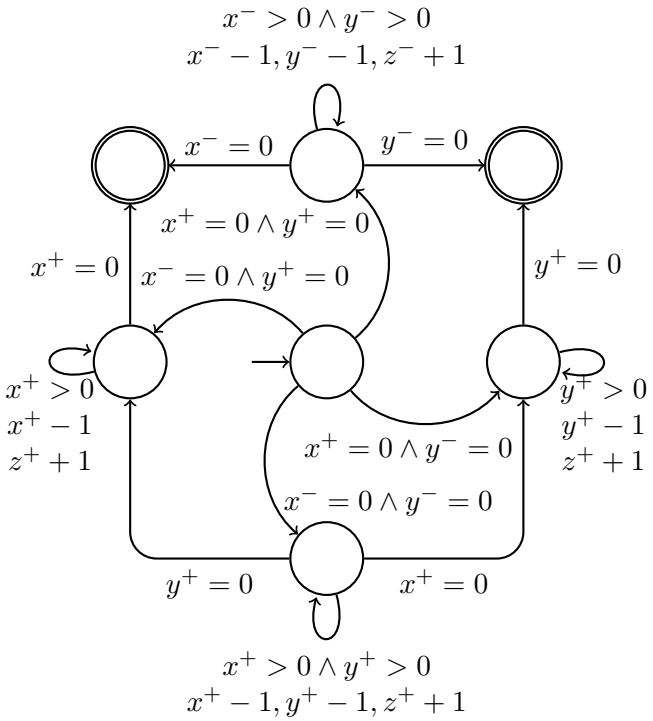


FIGURE 3.6 : $\mathcal{M}_{max}(x, y, z) = z := \max\{x, y\}$

Furthermore, all counters are increasing only in gadgets (or eventually normalized at the end) and decreasing only after. In addition, the total number of state is linear with the size of the expression because there is one gadget per node in the semantic tree. □

3.2 Complexity of decision problems with an expression combiner

Now we prove in the next lemma, that all decision problems can be solved by the zero-emptiness problems. Thus, we show different complexity cases for the zero-emptiness problems.

Lemma 3.3

All decision problems for functions defined by composition of functional $\mathbb{Z}^k$Sum-WA and an expression can be solved with constant Boolean combinations of zero-emptiness problems.

Proof

Given $\mathcal{F}$ a $\mathbb{Z}^n$Sum-WA, $\mathcal{G}$ a $\mathbb{Z}^m$Sum-WA, two expressions E_f over X_f and E_g over X_g such that $X_f \cap X_g = \varnothing$, $|X_f| \leq n$ and $|X_g| \leq m$. We also take a threshold $c \in \mathbb{Z}$, $\asymp \in \{\leq, <, =, >, \geq\}$ and $\succ \in \{>, \geq\}$.

Emptiness : $\exists u \in \mathcal{L}_{\mathcal{F}}.\ E_f([\![\mathcal{F}]\!](u)) \asymp c \iff \exists u \in \mathcal{L}_{\mathcal{A}}.\ E([\![\mathcal{A}]\!](u)) \succ 0$ with :

- $E := E_f + (-x_c)$ with x_c a fresh variable.
- $\mathcal{A}$ is like the automaton $\mathcal{F}$ which an additional vector dimension. All transitions are lifted with a zero on the $(n+1)^{\text{th}}$ field. To assign the last field to the value of c, we simulate a starting transition with a copy of the initial state (same outgoing transitions) and obviously, the $(n+1)^{\text{th}}$ field add c. The initial state of $\mathcal{A}$ is the new one. Remark, $\mathcal{L}_{\mathcal{F}} = \mathcal{L}_{\mathcal{A}}$.

Universality : The universality problems can be expressed by an emptiness negations. We define the operator negation **not** $\{\leq \mapsto >, < \mapsto \geq, > \mapsto \leq, \geq \mapsto <\}$.

$\boxed{\asymp \in \{\leq, <, >, \geq\}}$ $\forall u \in \mathcal{L}_{\mathcal{F}}.\ E_f([\![\mathcal{F}]\!](u)) \asymp c \iff \neg \exists u \in \mathcal{L}_{\mathcal{F}}.\ E_f([\![\mathcal{F}]\!](u))\ \mathbf{not}(\asymp)\ c$

$\boxed{\asymp \in \{=\}}$ $\forall u \in \mathcal{L}_{\mathcal{F}}.\ E_f([\![\mathcal{F}]\!](u)) = c \iff \neg\left(\exists u \in \mathcal{L}_{\mathcal{F}}.\ E_f([\![\mathcal{F}]\!](u)) < (\asymp)c \vee E_f([\![\mathcal{F}]\!](u)) > c\right)$

Comparison : $\forall u \in \mathcal{L}_{\mathcal{F}} \cap \mathcal{L}_{\mathcal{G}}.\ E_f([\![\mathcal{F}]\!](u)) \succ E_g([\![\mathcal{G}]\!](u)) \iff \exists u \in \mathcal{L}_{\mathcal{A}}.\ E(\mathcal{A}(u)) \succ 0$ with

- $E := E_f + (-E_g)$.
- $\mathcal{A} := \mathcal{F} \times \mathcal{G}$ the direct product. Remark, $\mathcal{L}_{\mathcal{F}} \cap \mathcal{L}_{\mathcal{G}} = \mathcal{L}_{\mathcal{A}}$. □

Theorem 3.4

Let $\mathcal{M}$ be a k-counter machine with m transitions, one reversal. $\mathcal{L}^0_{\mathcal{M}} \neq \varnothing$ if and only if $\mathcal{M}$ accepts some input in (km^{kC}) transitions where C is constant. ADMITTED [17]

Theorem 3.5

All decision problems are in PSPACE for functions defined by the composition of a functional $\mathbb{Z}^k$Sum-WA and an expression (even if the automaton is given as a product of k $\mathbb{Z}$Sum-WA). The problems are in NLOGPSPACE if k and the size of the expression are fixed with a unary encoding of weighs).

Proof

Given $\mathcal{A}$ a functional $\mathbb{Z}^k$Sum-WA with ℓ the maximal weight, m its number of transitions and E be an expression over X_E such that $|X_E| \leq k$. We show that $\exists u \in \mathcal{L}_{\mathcal{A}}.\ E(\mathcal{A}(u)) \asymp 0$ with $\asymp \in \{\leq, <, =, >, \geq\}$ is in PSPACE. By LEMMA 3.1 one can construct a reversal bounded counter machines $\mathcal{M}_{\mathcal{A}}$ which have $\mathcal{O}(mk\ell)$ transitions and $2k$ counters (increasing only). Moreover, by LEMMA 3.2 one can construct a reversal bounded counter machine $\mathcal{M}_E$ which has $\mathcal{O}(|E|)$ transitions, $2|E|$ counters and a single reversal (increasing-decreasing). Finally, we construct $\mathcal{M} := \mathcal{M}_{\mathcal{A}} \cdot \mathcal{M}_E \cdot \mathcal{M}_{\asymp}$ with $\mathcal{O}(m|E|\ell)$ transitions, $2|E|$ counters ($\mathcal{M}_{\mathcal{A}}$ use variables counter of $\mathcal{M}_E$, $\mathcal{M}_{\asymp}$ use the root counter of $\mathcal{M}_E$) and a single reversal (because $\mathcal{M}_{\mathcal{A}}$ is increasing only, $\mathcal{M}_E$ is increasing-decreasing and $\mathcal{M}_{\asymp}$ is decreasing only).

Then, $\mathcal{L}_{\mathcal{M}} \neq \varnothing \iff \exists u \in \mathcal{L}_{\mathcal{A}}.\ E(\mathcal{A}(u)) \asymp 0$. By THEOREM 3.4, $\mathcal{L}_{\mathcal{M}} \neq \varnothing$ if and only if $\mathcal{M}$ accepts an input in $\mathcal{O}\left((m|E|\ell)^{(4k)}\right)$ transitions. Thus, one can construct a non-deterministic space bounded Turing machine with a binary counter which guesses a run in $\mathcal{M}$ smaller than $\mathcal{O}\left((m|E|\ell)^{(4k)}\right)$. This machine accepts if $\mathcal{M}$ accepts, and rejects if the counter value exceeds the transition threshold value or if $\mathcal{M}$ rejects by transition default. Therefore, the Turing Machine uses $\mathcal{O}(k \log_2(m|E|\ell))$ space.

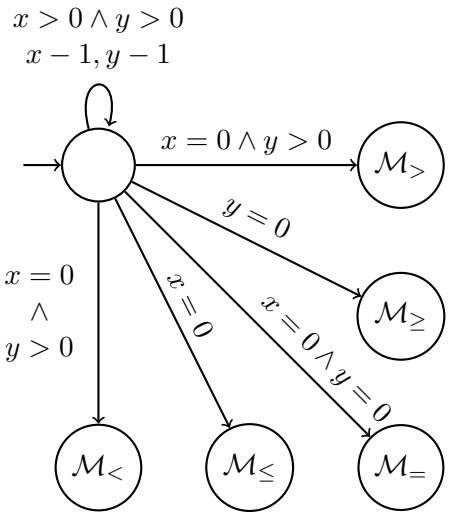


FIGURE 3.7 : Set one accepting state

Finally, by LEMMA 3.3 all other decision problems can be expressed with constant Boolean combinations of zero-emptiness problems. If the automaton is given by k automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ whose product defines $\mathcal{A}$, then its number of transitions must be m^k and the maximal value of a counter $\mathcal{O}\left((m|E|\ell)^{(8k)}\right)$. Thus, the complexity does not change. Remark, if k, $|E|$ are fixed and a unary encoding is used for weights (threshold included) the complexity would be in NLOGSPACE. □

Theorem 3.6

Zero-Emptiness problems are PSPACE-HARD for functions defined by composition of the product of k $\mathbb{Z}$Sum-WA and an expression (even with a unary encoding of weighs).

Proof

By reduction from the *Boolean Emptiness for intersection of regular languages* problem. Recall, $\bigcap_{i=1}^k \mathcal{L}_i \neq \varnothing$ where $\mathcal{L}_i$ be a regular language for each $i \in [1..k]$, is PSPACE-COMPLETE [25]. Given a set of k regular automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ one can construct k $\mathbb{Z}$Sum-WA $\widetilde{\mathcal{A}_1}, \ldots, \widetilde{\mathcal{A}_k}$ (put zero on all transitions). Remark, weights and threshold representation have the same size with a unary and binary encodings. For zero-emptiness problem with $\asymp \in \{\leq, =$

$, \geq\}$ we take the expression which just sum all variables, with $<$ subtract 1 at this sum and with $>$ add 1 at this sum. The reduction is lineare because the automata constructions are constant and the expressions constructions are lineare.

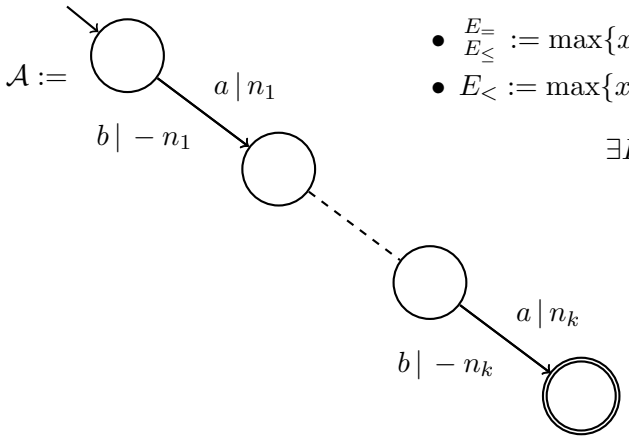
$$\begin{aligned}
\exists u \in \mathcal{L}_{\mathcal{A}_1} \cap \cdots \cap \mathcal{L}_{\mathcal{A}_k} &\iff \exists u \in \widetilde{\mathcal{A}_1} \cap \cdots \cap \widetilde{\mathcal{A}_k}.\ \sum_{i=1}^{k} [\![\mathcal{A}_i]\!](u) \asymp 0 \text{ where } \asymp \in \{\leq, =, \geq\} \\
&\iff \exists u \in \widetilde{\mathcal{A}_1} \cap \cdots \cap \widetilde{\mathcal{A}_k}.\ \sum_{i=1}^{k} [\![\mathcal{A}_i]\!](u) - 1 < 0 \\
&\iff \exists u \in \widetilde{\mathcal{A}_1} \cap \cdots \cap \widetilde{\mathcal{A}_k}.\ \sum_{i=1}^{k} [\![\mathcal{A}_i]\!](u) + 1 > 0
\end{aligned}$$ □

Theorem 3.7

Zero-Emptiness problems are NP-Hard for functions defined by composition of $\mathbb{Z}^k$Sum-WA and an expression (even when k and the size of the expressions are fixed).

Proof

By reduction from the *Set Partition* problem. Recall, given a multiset $S := \{n_1, \ldots, n_k\}$ of natural number, the question whether there exists I such that $\sum_{i\in I} n_i = \sum_{i\notin I} n_i$ is NP-complete [23, 28]. For any set S one can construct the $\mathbb{Z}$Sum-WA in figure 3.8 and expressions such that the (largest/smallest) difference between x_1 and x_2 should be exactly 0. Remark, k and $|E_{\asymp}|$ are fixed. The reduction is lineare because the automata constructions are lineare and the expressions constructions are constant.



- $\begin{matrix} E_= \\ E_\leq \end{matrix} := \max\{x_1 - x_2, x_2 - x_1\}$
- $E_\geq := \min\{x_1 - x_2, x_2 - x_1\}$
- $E_< := \max\{x_1 - x_2, x_2 - x_1\} - 1$
- $E_> := \min\{x_1 - x_2, x_2 - x_1\} + 1$

$$\begin{aligned}
\exists I.\ \sum_{i\in I} n_i = \sum_{i\notin I} n_i &\iff \exists u \in \{a,b\}^k.\ \sum_{u[i]=a} n_i = \sum_{u[i]=b} n_i \\
&\iff \exists u \in \{a,b\}^k, \exists x_1, x_2.\ \bigwedge \begin{pmatrix} x_1 = \sum_{u[i]=a} n_i \\ x_2 = \sum_{u[i]=b} n_i \\ E_{\asymp}(x_1, x_2) \geq 0 \end{pmatrix} \\
&\iff \exists u \in \{a,b\}^k.\ E_{\asymp}([\![\mathcal{A}]\!](u)) \asymp 0 \text{ where } \asymp \in \{<, \leq, =, \geq, >\}
\end{aligned}$$

figure 3.8 : a to put $n \in \{n_1, \ldots n_k\}$ in I, b otherwise □

Theorem 3.8

Zero-Emptiness problems are NP-Hard for function defined by composition of $\mathbb{Z}^k$Sum-WA and an expression (even with unary encoding of weights, but k and the size of the expression cannot both be fixed).

Proof

By reduction from *3-Partition* problem. Recall, given a multiset $S := \{n_1, \ldots, n_{3k}\}$ of integers encoded in unary, the question whether exists $I_1, \ldots I_k$ a partition of S such that $\sum_{n\in I_1} n = \cdots = \sum_{n\in I_k} n$ and $|I_1| = \cdots = |I_k| = 3$ is Strongly NP-Complete [16, 28]. For any set S one can construct the $\mathbb{Z}^{2k}$Sum-WA in figure 3.10 and the expressions in figure 3.9. Remark, on the one hand $\mathcal{A}$ evaluates $2k$ variables, the k first represent respective sum of I_j's elements for each $j \in \{1, \ldots, k\}$ and the k last represent respective cardinal of I_j for each $j \in \{1, \ldots, k\}$. On the other hand, partition of S by $I_1, \ldots, I_k$ is fulfilled by construction of $\mathcal{A}$. The reduction is lineare because the automata and expressions constructions are lineare.

- $\begin{matrix} E_= \\ E_\leq \end{matrix} := \max \begin{Bmatrix} \min\{x_1, \ldots, x_k\} - \max\{x_1, \ldots, x_k\} \\ \max\{x_1, \ldots, x_k\} - \min\{x_1, \ldots, x_k\} \\ \min\{x'_1, \ldots, x'_k, 3\} - \max\{x'_1, \ldots, x'_k, 3\} \\ \max\{x'_1, \ldots, x'_k, 3\} - \min\{x'_1, \ldots, x'_k, 3\} \end{Bmatrix}$
- $E_\geq := \min \begin{Bmatrix} \min\{x_1, \ldots, x_k\} - \max\{x_1, \ldots, x_k\} \\ \max\{x_1, \ldots, x_k\} - \min\{x_1, \ldots, x_k\} \\ \min\{x'_1, \ldots, x'_k, 3\} - \max\{x'_1, \ldots, x'_k, 3\} \\ \max\{x'_1, \ldots, x'_k, 3\} - \min\{x'_1, \ldots, x'_k, 3\} \end{Bmatrix}$
- $E_< := \max \begin{Bmatrix} \min\{x_1, \ldots, x_k\} - \max\{x_1, \ldots, x_k\} \\ \max\{x_1, \ldots, x_k\} - \min\{x_1, \ldots, x_k\} \\ \min\{x'_1, \ldots, x'_k, 3\} - \max\{x'_1, \ldots, x'_k, 3\} \\ \max\{x'_1, \ldots, x'_k, 3\} - \min\{x'_1, \ldots, x'_k, 3\} \end{Bmatrix} - 1$
- $E_> := \min \begin{Bmatrix} \min\{x_1, \ldots, x_k\} - \max\{x_1, \ldots, x_k\} \\ \max\{x_1, \ldots, x_k\} - \min\{x_1, \ldots, x_k\} \\ \min\{x'_1, \ldots, x'_k, 3\} - \max\{x'_1, \ldots, x'_k, 3\} \\ \max\{x'_1, \ldots, x'_k, 3\} - \min\{x'_1, \ldots, x'_k, 3\} \end{Bmatrix} + 1$

figure 3.9 : The (largest/smallest) difference between the sums or the cardinals & 3 should be exactly 0

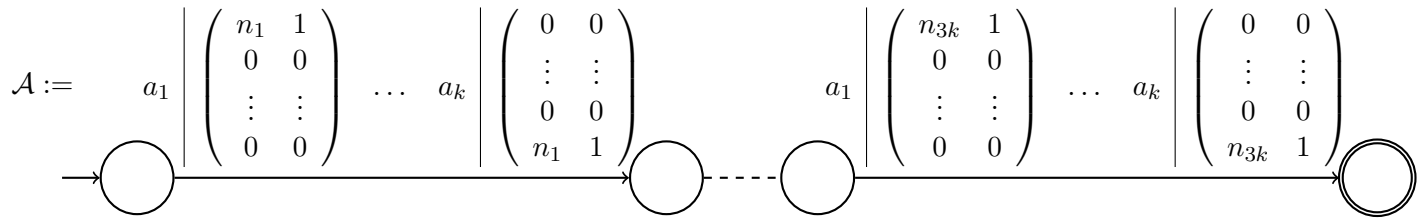


FIGURE 3.10 : The letter a_j to put $n \in \{n_1, \dots, n_{3k}\}$ in the set I_j (matrix form is just of readability)

$$\begin{aligned}
&\exists I_1, \dots, I_k. \sum_{n \in I_1} n = \cdots = \sum_{n \in I_k} n \wedge |I_1| = \cdots = |I_k| \\
&\quad \iff \exists u \in \{a_1, \dots, a_k\}^{3k}. \sum_{u[i]=a_1} n_i = \cdots = \sum_{u[i]=a_k} n_i \wedge \forall j \in [1..k].\ |u|_{a_j} = 3 \\
&\quad \iff \exists u \in \{a_1, \dots, a_k\}^{3k}. \exists x_1 \dots x_k, x'_1 \dots x'_k. \bigwedge_{j=1}^{k} x_j = \sum_{u[i]=a_j} n_i \wedge \bigwedge_{j=1}^{k} x'_j = |u|_{a_j} \wedge E_{\asymp}(x_1, \dots, x_k, x'_1, \dots, x'_k) \asymp 0 \\
&\quad \iff \exists u \in \{a_1, \dots, a_k\}^{3k}. E_{\asymp}([\![\mathcal{A}]\!](u)) \asymp 0 \text{ where } \asymp \in \{<, \leq, =, \geq, >\}
\end{aligned}$$ □

Theorem 3.9

Zero-Emptiness problems are NLogSpace-Hard for function defined by composition of $\mathbb{Z}^k$Sum-WA and an expression (even if k and the size of the expression are fixed with a unary encoding of weighs).

Proof

By reduction from the *membership of non-deterministic regular automata* problem. Recall, given $\mathcal{A} \in REG_\Sigma$ a regular automata, deciding $\exists u \in \Sigma^*.\ u \in \mathcal{L}_\mathcal{A}$ is NLogSpace-Complete [22, 30]. Given $\mathcal{A} \in REG_\Sigma$, one can construct a $\mathbb{Z}$Sum-WA $\widetilde{\mathcal{A}}$ (put zero on all transitions) and an expression E which just returns immediately the value of the unique variable. Remark, weights and threshold representation have the same size with unary and binary encodings, k and $|E|$ are fixed. The reduction is constant because the automata and expressions constructions are constant.

$$\exists u \in \mathcal{L}_\mathcal{A} \iff \exists u \in \mathcal{L}_{\widetilde{\mathcal{A}}}.\ [\![\widetilde{\mathcal{A}}]\!](u) \asymp 0 \text{ where } \asymp \in \{\leq, <, =, >, \geq\}$$ □

3.3 Summary

$\Sigma^* \to \mathbb{Z}^k$	$\mathbb{Z}^k \to \mathbb{Z}$	fixed parameters	encoding	$\exists$	$\forall$	$\succ$
functional $\mathbb{Z}^k$Sum-WA	Expr		binary	NP -Hard	CoNP-Hard	
			unary	NP-Hard	CoNP-Hard	
		$k, \lvert E\rvert$	binary	NP-Hard	CoNP-Hard	
			unary	NLogSpace − Complete		
functional $\prod^k \mathbb{Z}$Sum-WA	Expr		binary	PSpace − Complete		
			unary			
		$k, \lvert E\rvert$	binary	NP-Hard	CoNP-Hard	
			unary	NLogSpace − Complete		

4 Presburger

In this section, we extend expression combiners to existential Presburger logic and show several translations.

Definition : Presburger arithmetic

An existential Presburger formula is given by the following grammars where variables are in $\mathbb{Z}$.

$$\phi := t = t \mid t < t \mid \phi \vee \phi \mid \phi \wedge \phi \mid \exists x.\ \phi \qquad\qquad t := 0 \mid 1 \mid x \mid t + t$$

A formula is associated with its semantics tree (described beside for $\phi := \exists x.\ x + 0 = 1$). We define the language $\mathcal{T}(\phi) \subseteq \{\alpha, \beta\}^*$ to describe all correct paths in the semantics tree of ϕ. Let's us note ϕ_u to describe the sub-formula, or t_u for terms, at position $u \in \mathcal{T}(\phi)$. We define the proposition $\mathbf{Var}(\phi_u)$ to hold if ϕ_u is a variable. The size of a formula ϕ, is $|\phi| := |\mathcal{T}(\phi)|$ and the set of ϕ's free variables are denoted $\mathbf{Free}(\phi)$.

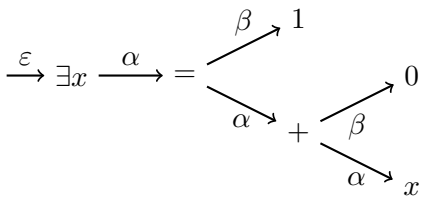


FIGURE 4.1 : Semantic tree

Traditionally, the Presburger arithmetic is given with variables interpreted over natural numbers, but both models have the same expressiveness (see technical details in APPENDIX B). Remark, one can represent a constant by a formula with the same size as its encoding. For a binary encoding of C we use $\exists x_{log_2(c)}, \dots, x_1.\ x_{log_2(c)} + x_{log_2(c)} + ... + x_1 + x_1$ and for a unary encoding we use $\underbrace{1 + \cdots + 1}_{C}$.

4.1 $\mathbb{Z}^k$Sum-WA valuation to Presburger combiner

We propose a translation of $\mathbb{Z}^k$Sum-WA to existential Presburger formulas through a grammar which uses an external theorem. This will allow us to reduce all decision problem to the satisfiability of the formula.

Theorem 4.1

Satisfiability problem for an existential Presburger formula is NP-COMPLETE. ADMITTED [33]

Definition : Context-Free Grammar

The syntax of a context-free grammar is the tuple $\langle \mathcal{V}, \Sigma, P, S \rangle$ where :

- $\mathcal{V}$ non-terminal alphabet
- Γ terminal alphabet
- $P \subseteq \mathcal{V} \times (\mathcal{V} \cup \Sigma)^*$ rewrite rules
- S start symbol

Given $u, v \in (\mathcal{V} \cup \Gamma)^*$, we write $u \rightarrowtail v$ if there exist $u_1, u_2, u' \in (\mathcal{V} \cup \Gamma)^*$ and $A \in \mathcal{V}$ such that $u = u_1 A u_2$, $v = u_1 u' u_2$ and $P(A, u')$. We denote by $\rightarrowtail^*$ the reflexive transitive closure of $\rightarrowtail$. The language of $\mathcal{G}$ is $\mathcal{L}_{\mathcal{G}} := \{w \in \Gamma^* : S \rightarrowtail^* w\}$.

Lemma 4.2

Let $\mathcal{A}$ be a functional $\mathbb{Z}^k$Sum-WA and $\Gamma := \{\alpha_1^-, \alpha_1^+, \dots, \alpha_k^-, \alpha_k^+\}$ a fixed alphabet of terminal symbols. One can compute, in linear time, a context free grammar $\mathcal{G}$ such that :

$$w \in \mathcal{L}_{\mathcal{G}} \iff \exists u \in \mathcal{L}_{\mathcal{A}}.\ \forall i \in [1..k].\ [\![\mathcal{A}]\!](u)[i] = |w|_{\alpha_i^+} - |w|_{\alpha_i^-}$$

Proof

Basically, we construct a grammar whose non-terminal alphabet is a state of $\mathcal{A}$ and its production rules are transitions. But, if weights of the automaton have binary encoding then we must use a trick to avoid an exponential blow-up. To do this, we simply add some special non-terminal symbols and add the production rules, see FIGURE 4.2. Then, to produce x times the symbol $\alpha_j^{\bowtie}$ for some $j \in [1..k]$ and $\bowtie \in \{+, -\}$, we decompose x into a sum or substraction of a power 2 with non-terminal symbols $[\bowtie 2_j]$. For states p, q and letter a, we note the translation weights as follow.

$$\langle \gamma(p,a,q) \rangle := \left(\alpha_1^{\mathbf{Sign}(\gamma(p,a,q)[1])}\right)^{|\gamma(p,a,q)[1]|} \dots \left(\alpha_k^{\mathbf{Sign}(\gamma(p,a,q)[k])}\right)^{|\gamma(p,a,q)[k]|}$$

Finally, given $\mathcal{A} := \langle \Sigma, Q, s, F, \Delta, \gamma, \ell \rangle$ we construct $\mathcal{G} := \{\mathcal{V}, \Gamma, P, S\}$ where :

- $\mathcal{V} := Q \cup \bigcup_{i=0}^{\lfloor log_2(\ell) \rfloor} \bigcup_{j=0}^{k} \{\left[+2_j^i\right], \left[-2_j^i\right]\}$
- $\Gamma := \bigcup_{j=0}^{k} \{\alpha_j^+, \alpha_j^-\}$
- $S := s$
- $P := \bigcup \left(\bigcup_{(p,a,q) \in \Delta} \{p \rightarrowtail q \langle \gamma(p,a,q) \rangle\} \cup \bigcup_{p \in F} \{p \rightarrowtail \varepsilon\} \right)$

$$\forall i \in [1..k].\ \begin{cases} \left[+2_i^0\right] \rightarrowtail \alpha_i^+ \\ \vdots \\ \left[+2_i^{\lfloor log_2(\ell) \rfloor}\right] \rightarrowtail \left[+2_i^{\lfloor log_2(\ell) \rfloor - 1}\right] \left[+2_i^{\lfloor log_2(\ell) \rfloor - 1}\right] \end{cases}$$

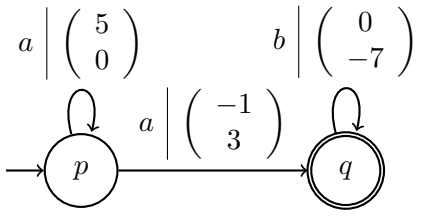
$$\forall i \in [1..k].\ \begin{cases} \left[-2_i^0\right] \rightarrowtail \alpha_i^- \\ \vdots \\ \left[-2_i^{\lfloor log_2(\ell) \rfloor}\right] \rightarrowtail \left[-2_i^{\lfloor log_2(\ell) \rfloor - 1}\right] \left[-2_i^{\lfloor log_2(\ell) \rfloor - 1}\right] \end{cases}$$

FIGURE 4.2 : weights encoding

Obviously, there is a bijection between the derivations of $\mathcal{G}$ and paths in $\mathcal{A}$. Indeed, all rewriting rules consume a non-terminal symbol of Q and produce one of them (possibly zero if the consumed state is accepting). In particular, we have $w \in \mathcal{L}_{\mathcal{G}} \iff s \rightarrowtail^* q_f \in F \rightarrowtail w \not\rightarrowtail$ and then all words of $\mathcal{L}_{\mathcal{G}}$ are in bijection with an accepting path of $\mathcal{A}$. Furthermore, the derivation of all extra non-terminal symbol is deterministic. By a straightforward induction we can obtain the correction of weight translations along a path. Thus, the size of $\mathcal{G}$ is clearly $\mathcal{O}(|\mathcal{A}|)$. □

Example

$\mathcal{A} :=$



$\mathcal{G} := \{\mathcal{V}, \Gamma, P, S\}$

- $S := p$
- $\mathcal{V} := \{p, q\} \cup \bigcup_{i=0}^{\lfloor log_2(7) \rfloor} \bigcup_{j=0}^{k} \{\left[+2_j^i\right], \left[-2_j^i\right]\}$
- $\Gamma := \bigcup_{j=0}^{k} \{\alpha_j^+, \alpha_j^-\}$
- $P := \left\{ \begin{array}{l} p \rightarrowtail \left[+2_1^2\right]\left[+2_1^0\right] p \\ p \rightarrowtail \left[-2_1^0\right]\left[+2_2^1\right]\left[+2_2^0\right] q \\ q \rightarrowtail \left[-2_2^2\right]\left[-2_2^1\right]\left[-2_2^0\right] q \\ q \rightarrowtail \varepsilon \end{array} \right\}$

Definition : Parikh Image

The Parikh image of a string is the function that maps each symbol of the alphabet to the number of its occurrences in the string. The Parikh image of a language is the set of Parikh images of its words.

Theorem 4.3

Given a context-free grammar $\mathcal{G}$ on terminals symbols $a_1, \ldots, a_k$, one can compute an existential Presburger formula $\phi_{\mathcal{G}}$ for the Parikh image of $\mathcal{L}_{\mathcal{G}}$ in linear time i.e. such that $\phi_{\mathcal{G}}(x_1, ..., x_k)$ holds if and only if some $w \in \mathcal{L}_{\mathcal{G}}$ contains a_j exactly x_j times for each $j \in [1..k]$. Admitted [37]

Theorem 4.4

Emptiness problems are NP-Complete, Comparison and Universality problems are CoNP-Complete for function defined by composition of $\mathbb{Z}^k$Sum-WA and existential Presburger formula.

Proof

Given $\mathcal{F}$ a $\mathbb{Z}^n$Sum-WA, $\mathcal{G}$ a $\mathbb{Z}^m$Sum-WA, two existential Presburger formulae ϕ_f, ϕ_g with respectively $n+1$, $m+1$ free variables, an integer $c \in \mathbb{Z}$, $\asymp \in \{\leq, <, =, >, \geq\}$ and $\succ \in \{>, \geq\}$. We define the operator negation **not** : $\{> \mapsto \leq, \geq \mapsto <\}$. By lemma 4.2 and theorem 4.3 one can construct, in linear time, an existential Presburber formula $\phi_{\mathcal{F}}$ such that $\phi_{\mathcal{F}}(x_1, \ldots, x_n)$ holds if and only if $\exists u \in \mathcal{L}_{\mathcal{F}}.\ \forall i \in [1..n].\ [\![\mathcal{F}]\!](u)[i] = x_i$. Same idea for $\phi_{\mathcal{G}}$ and $\phi_{\mathcal{F}\times\mathcal{G}}$ for the direct product $\mathcal{F} \times \mathcal{G}$.

Emptiness : The problem can be expressed with following formula and by theorem 4.1, its satisfiability is NP-Complete.

$$\exists x_1, \ldots, x_n, x \in \mathbb{Z}.\ \bigwedge \begin{pmatrix} \phi_{\mathcal{F}}(x_1, \ldots, x_n) \\ \phi_f(x_1, \ldots, x_n, x) \\ x \asymp c \end{pmatrix}$$

Universality : The problems can be expressed by emptiness negations.

Comparison : The problem can be expressed with the negation of the following formula and by theorem 4.1, its non-satisfiability is CoNP-Complete.

$$\exists \begin{pmatrix} x_1, \ldots, x_n \\ y_1, \ldots, y_m \\ x, y \end{pmatrix} \in \mathbb{Z}.\ \bigwedge \begin{pmatrix} \phi_{\mathcal{F}\times\mathcal{G}}(x_1, \ldots x_n, y_1, \ldots, x_m) \\ \phi_f(x_1, \ldots, x_n, x) \wedge \phi_g(y_1, \ldots, y_m, y) \\ x \prec y \end{pmatrix}$$

□

4.2 Translations from/to Presburger combiner

Now, we present a translation of an expression combiner to an existential Presburger formula, to update some the upper bound of decision problem in the previous section (when valuation is given by a $\mathbb{Z}^k$Sum-WA).

We give also, a translation of an existential Presburger formula to a reversal bounded counter machine for extend the PSPACE complexity result (when valuation is given by a product of k $\mathbb{Z}$Sum-WA). The last translation result is known because it is a consequence of the fact that the class of Presburger relations is exactly the class of relations computable by deterministic reversal fixed counter machine [18].

Lemma 4.5

Let E be an expression over $X := \{x_1, \ldots, x_k\}$ (defined in previous section). One can construct in linear time an existential Presburger formula ϕ_E with free variables $\{x_1, \ldots, x_k, x\}$ such that :

$$\forall \nu : X \to \mathbb{Z}.\ \nu \cup \{x \mapsto n\} \models \phi_E \iff E(\nu) = n$$

Proof

We show the construction of ϕ_E by structural induction on E. Let's note E_u to describe the sub-expression at position $u \in \mathcal{T}(E)$ in the semantics tree.

Constants : Case $E_u := C$ where $C \in \{0, 1\}$. $\phi_u(x_1, \ldots, x_k, x_u) := (x_u = C)$

Minus : Case $E_u := -E_{u\alpha}$. By induction hypothesis $\nu \cup \{x_{u\alpha} \mapsto n\} \models \phi_{u\alpha} \iff E_{u\alpha}(\nu) = n$.

$$\phi_u(x_1, \ldots x_k, x_u) := \exists x_{u\alpha}.\ \phi_{u\alpha}(x_1, \ldots, x_k, x_{u\alpha}) \wedge x_u + x_{u\alpha} = 0$$

Sum : Case $E_u := E_{u\alpha} + E_{u\beta}$. By induction hypothesis $\nu \cup \{x_{u\alpha} \mapsto n\} \models \phi_{u\alpha} \iff E_{u\alpha}(\nu) = n$ and $\nu \cup \{x_{u\beta} \mapsto m\} \models \phi_{u\beta} \iff E_{u\beta}(\nu) = m$.

$$\phi_u(x_1, \ldots x_k, x_u) := \exists x_{u\alpha}, x_{u\beta}.\ \phi_{u\alpha}(x_1, \ldots, x_k, x_{u\alpha}) \wedge \phi_{u\beta}(x_1, \ldots, x_k, x_{u\beta}) \wedge x_u = x_{u\alpha} + x_{u\beta}$$

Variable : Case $E_u := x_i$. $\phi_u(x_1, \ldots, x_k, x_u) := (x_u = x_i)$

Maximum : Case $E_u := \max\{E_{u\alpha}, E_{u\beta}\}$. By induction hypothesis $\nu \cup \{x_{u\alpha} \mapsto n\} \models \phi_{u\alpha} \iff E_{u\alpha}(\nu) = n$ and $\nu \cup \{x_{u\beta} \mapsto m\} \models \phi_{u\beta} \iff E_{u\beta}(\nu) = m$.

$$\phi_u(x_1, \ldots x_k, x_u) := \exists x_{u\alpha}, x_{u\beta}.\ \phi_{u\alpha}(x_1, \ldots, x_k, x_{u\alpha}) \wedge \phi_{u\beta}(x_1, \ldots, x_k, x_{u\beta}) \bigwedge \begin{pmatrix} x_u = x_{u\alpha} \vee x_u = x_{u\beta} \\ x_u \geq x_{u\alpha} \wedge x_u \geq x_{u\beta} \end{pmatrix}$$

Minimum : Case $E_u := \min\{E_{u\alpha}, E_{u\beta}\}$. By induction hypothesis $\nu \cup \{x_{u\alpha} \mapsto n\} \models \phi_{u\alpha} \iff E_{u\alpha}(\nu) = n$ and $\nu \cup \{x_{u\beta} \mapsto m\} \models \phi_{u\beta} \iff E_{u\beta}(\nu) = m$.

$$\phi_u(x_1, \ldots x_k, x_u) := \exists x_{u\alpha}, x_{u\beta}.\ \phi_{u\alpha}(x_1, \ldots, x_k, x_{u\alpha}) \wedge \phi_{u\beta}(x_1, \ldots, x_k, x_{u\beta}) \bigwedge \begin{pmatrix} x_u = x_{u\alpha} \vee x_u = x_{u\beta} \\ x_u \leq x_{u\alpha} \wedge x_u \leq x_{u\beta} \end{pmatrix}$$

□

Corollary 4.6

By LEMMA 4.5 and THEOREM 4.4, emptiness problems are NP-COMPLETE, comparison and universality problems are CONP-COMPLETE for functions defined by composition of $\mathbb{Z}^k$Sum-WA and an expression (defined in the previous section). This does not work when the valuation is given as a product of k $\mathbb{Z}$Sum-WA because all of them must read the same word which forces the computation of the product to construct the grammar.

Lemma 4.7

Given an existential Presburger formula ϕ, one can construct a $2|\phi|$-counter machine $\mathcal{M}$ whose $X := \bigcup_{u \in \mathcal{T}(\phi)} \{x_u^+, x_u^-\}$ its counter set (one reversal, increasing-decreasing) and with $\mathcal{O}(|\phi|)$ transitions such that :

$$\forall \nu : \mathbf{Free}(\phi) \to \mathbb{Z}.\ \nu \models \phi \iff \varepsilon \in \mathcal{L}^{\hat{\nu}}_{\mathcal{M}}$$

where $\forall u \in \mathcal{T}(E).\hat{\nu}(x_u^+) := \begin{cases} \nu(\phi_u) & \textbf{if } \phi_u \in \mathbf{Free}(\phi) \wedge 0 < \nu(\phi_u) \\ 0 & \textbf{otherwise} \end{cases}$ $\hat{\nu}(x_u^-) := \begin{cases} -\nu(\phi_u) & \textbf{if } \phi_u \in \mathbf{Free}(\phi) \wedge \nu(\phi_u) < 0 \\ 0 & \textbf{otherwise} \end{cases}$

Proof

We start by noting that the term grammar of a formula is included in the expression grammar (defined in the previous section). So, we will use the LEMMA 3.2 to build a machine which evaluates an expression and stores the

result in its root counter. We show the construction by structural induction on ϕ. Let's note ϕ_u to describe the sub-formula, or t_u for term, at position $u \in T(\phi)$ in semantic tree.

Compare : Case $\phi_u := t_{u\alpha} \asymp t_{u\beta}$ where $\asymp \in \{=, <\}$. By LEMMA 3.2 one can construct $\mathcal{M}_{u\alpha}$ over $X_{u\alpha} := \bigcup_{u\in\mathcal{T}(\phi_{u\alpha})}\{x^+, x^-\}$ and $\mathcal{M}_{u\beta}$ over $X_{u\beta} := \bigcup_{u\in\mathcal{T}(\phi_{u\beta})}\{x^+, x^-\}$ denotes respectively $\phi_{u\alpha}$ with the root couple $(x^-_{u\alpha}, x^+_{u\alpha})$ and $\phi_{u\beta}$ with the root couple $(x^-_{u\beta}, x^+_{u\beta})$. After the computation of $\mathcal{M}_{u\alpha}$ and $\mathcal{M}_{u\beta}$, one proceeds to test $(x^+_{u\alpha} + x^-_{u\beta}) \asymp (x^+_{u\beta} + x^-_{u\alpha})$ realized by $\mathcal{M}_{\text{add}}$ and $\mathcal{M}_{\asymp}$, see FIGURE 4.3 and FIGURE 3.7 (just for $<$ and $=$). To do this, we construct $\mathcal{M}_u := \mathcal{M}_{u\alpha} \cdot \mathcal{M}_{u\beta} \cdot \mathcal{M}_{\text{add}}(x^+_{u\alpha}, x^-_{u\beta}) \cdot \mathcal{M}_{\text{add}}(x^+_{u\beta}, x^-_{u\alpha}) \cdot \mathcal{M}_{\asymp}(x^+_{u\alpha}, x^+_{u\beta})$.

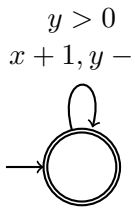


FIGURE 4.3 : $\mathcal{M}_{\text{add}}$

Or : Case $\phi_u := \phi_{u\alpha} \vee \phi_{u\beta}$. By induction hypothesis $\mathcal{M}_{u\alpha}$ over $X_{u\alpha} := \bigcup_{u\in\mathcal{T}(\phi_{u\alpha})}\{x^+, x^-\}$ and $\mathcal{M}_{u\beta}$ over $X_{u\beta} := \bigcup_{u\in\mathcal{T}(\phi_{u\beta})}\{x^+, x^-\}$ denotes respectively $\phi_{u\alpha}$ and $\phi_{u\beta}$. The machine choose non-deterministically one of two formulas. To do this, we construct $\mathcal{M}_u := \mathcal{M}_{u\alpha} \parallel \mathcal{M}_{u\beta}$. Remark, $X_{u\alpha} \cap X_{u\beta} = \varnothing$ and then $\parallel$ is well define.

And : Case $\phi_u := \phi_{u\alpha} \wedge \phi_{u\beta}$. By induction hypothesis $\mathcal{M}_{u\alpha}$ over $X_{u\alpha} := \bigcup_{u\in\mathcal{T}(\phi_{u\alpha})}\{x^+, x^-\}$ and $\mathcal{M}_{u\beta}$ over $X_{u\beta} := \bigcup_{u\in\mathcal{T}(\phi_{u\beta})}\{x^+, x^-\}$ denotes respectively $\phi_{u\alpha}$ and $\phi_{u\beta}$. The machine control that $\mathcal{M}_{u\alpha}$ and $\mathcal{M}_{u\beta}$ accept ε. To do this, we construct $\mathcal{M}_u := \mathcal{M}_{u\alpha} \cdot \mathcal{M}_{u\beta}$.

Exists : Case $\phi_u := \exists x.\ \phi_{u\alpha}$. By induction hypothesis $\mathcal{M}_{u\alpha}$ over $X_{u\alpha} := \bigcup_{u\in\mathcal{T}(\phi_{u\alpha})}\{x^+, x^-\}$ denotes $\phi_{u\alpha}$. Before the computation of $\mathcal{M}_{u\alpha}$, the machine guess a value for variable x realized by $\mathcal{M}_{\exists}$, see FIGURE 4.4. To do this, we construct $\mathcal{M}_u := \mathcal{M}_{\exists} \cdot \mathcal{M}_{u\alpha}$.

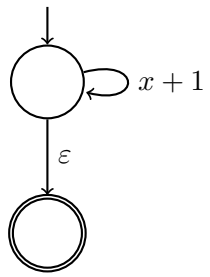


FIGURE 4.4 : $\mathcal{M}_{\exists}$ □

Corollary 4.8

By LEMMA 4.7 and THEOREM 3.5, all decision problems are in PSPACE for functions defined by the composition of a functional $\mathbb{Z}^k$Sum-WA and an existential Presburger formula (even if the automaton is given as a product of k $\mathbb{Z}$Sum-WA).

4.3 Summary

$\Sigma^* \to \mathbb{Z}^k$	$\mathbb{Z}^k \to \mathbb{Z}$	fixed parameters	encoding	$\exists$	$\forall$	$\succ$
functional $\mathbb{Z}^k$Sum-WA	Expr		binary	NP − COMPLETE	CoNP − COMPLETE	
			unary	NP − COMPLETE	CoNP − COMPLETE	
		$k, \lvert E\rvert$	binary	NP − COMPLETE	CoNP − COMPLETE	
			unary	NLOGSPACE − COMPLETE		
	Presburger		binary	NP−COMPLETE	CoNP−COMPLETE	
			unary			
functional $\prod^k \mathbb{Z}$Sum-WA	Expr		binary	PSPACE − COMPLETE		
			unary			
		$k, \lvert E\rvert$	binary	NP − COMPLETE	CoNP − COMPLETE	
			unary	NLOGSPACE − COMPLETE		
	Presburger		binary	PSPACE − COMPLETE		
			unary			

5 Robustness

Most computer systems are integrated in a physical environment and the data processed often contains a small amount of errors or uncertainty (for example, the data sensor acquisition in reactive systems for simultaneous localization and mapping). Other examples are text editors confronted with possibly misspelled keywords and DNA chains sometimes incorrectly sequenced in computational biology. It is desirable that such random noise in the input does not cause an unpredictable behavior of the system, overlooked sensitivity to disturbance or inaccuracy of individually considered image. In this section, we prove that the lipschitz robustness problem is undecidable in general for functions and distances realized by a weighted automata. Nevertheless, it becomes decidable if the function and the distance are closed under max, plus, minus operations and when the comparison problems are decidable. In addition, we define a class of functions called "good for distance"and we show that it is decidable whether those functions realize a distance.

Definition : Metric space

A metric space is a tuple (S, d) where S is a set and $d : S \times S \to \mathbb{R}$ which satisfies following axioms.

non-negative : $\forall u, v \in S.\ d(u, v) \geq 0$

symmetry : $\forall u, v \in S.\ d(u, v) = d(v, u)$

separation : $\forall u, v \in S.\ d(u, v) = 0 \leftrightarrow u = v$

triangle inequality : $\forall u, v, w \in S.\ d(u, v) + d(v, w) \geq d(u, w)$

Definition : K-Lipschitz robustness

Given $K > 0$, a metric spaces (S, d) and $f : S \to \mathbb{R}$ with $dom(f)^2 \subseteq dom(d)$. We say that f is K-robust if it satisfies :

$$\forall u, v \in dom(f).\ |f(u) - f(v)| \leq K \times d(u, v)$$

This notion is not appropriate in the discrete setting, as discrete functions are in general not continuous.

Definition : Word convolution

To define a notion of distance between words, we need automata with many parameters. A technique is to encode all words into a single one. To simplify the parameter process, one can find in the literature the convention that stalling words on left.

Let Σ and Γ be two alphabets, we set $\Sigma \otimes \Gamma := (\Sigma \times \Gamma) \cup (\{\#\} \times \Gamma) \cup (\Sigma \times \{\#\})$. Now we can define a word in $(\Sigma \otimes \Gamma)^*$ as follows.

$$\forall u \in \Sigma^*, \forall v \in \Gamma^*, u \otimes v := \begin{cases} \varepsilon & \text{if } u = \varepsilon \wedge v = \varepsilon \\ (a, \#) \cdot u' \otimes \varepsilon & \text{if } u = au' \wedge v = \varepsilon \text{ where } a \in \Sigma.\ u' \in \Sigma^* \\ (\#, v') \cdot \varepsilon \otimes v' & \text{if } v = \varepsilon \wedge v = bv' \text{ where } a \in \Gamma.\ v' \in \Gamma^* \\ (a, b) \cdot u' \otimes v' & \text{if } u = au' \wedge v = av' \text{ where } a \in \Sigma.\ b \in \Gamma.\ u' \in \Sigma^*.\ v' \in \Gamma^* \end{cases}$$

The size of $u \otimes v$ is obviously $|u \otimes v| := \max\{|u|, |v|\}$. In the remainder, we use some operations, adding useless extra words and permute parameters. To do this, we just apply the respective transformation on all transitions of the WA.

Definition : Domain restriction

Given $\mathcal{A} := \langle \Sigma, Q_{\mathcal{A}}, s_{\mathcal{A}}, F_{\mathcal{A}}, \Delta_{\mathcal{A}}, \gamma_{\mathcal{A}} \rangle$ be a $\mathbb{Z}^k$Sum-WA and a regular language defined $\mathcal{B} := \langle \Sigma, Q_{\mathcal{B}}, s_{\mathcal{B}}, F_{\mathcal{B}}, \Delta_{\mathcal{B}} \rangle$ a regular automata. We describe the construction of the domain restriction $[\mathcal{A}]_{\mathcal{L}_{\mathcal{B}}} := \langle \Sigma, Q_{\mathcal{A}} \times Q_{\mathcal{B}}, (s_{\mathcal{A}}, s_{\mathcal{B}}), F_{\mathcal{A}} \times F_{\mathcal{B}}, \Delta, \gamma \rangle$ such that $\mathcal{L}_{[\mathcal{A}]_{\mathcal{L}_{\mathcal{B}}}} = \mathcal{L}_{\mathcal{B}} \wedge \forall u \in \mathcal{L}_{[\mathcal{A}]_{\mathcal{L}_{\mathcal{B}}}}.\ \left[\!\left[[\mathcal{A}]_{\mathcal{L}_{\mathcal{B}}} \right]\!\right](u) = [\![\mathcal{A}]\!](u)$ and where Δ and γ as follow.

$$p_1 \xrightarrow{a \mid v}_{\mathcal{A}} q_1 \wedge p_2 \xrightarrow{a}_{\mathcal{B}} q_2 \Rightarrow (p_1, p_2) \xrightarrow{a \mid v}_{[\mathcal{A}]_{\mathcal{L}_{\mathcal{B}}}} (q_1, q_2)$$

5.1 Undecidability of robustness

We prove the undecidability of the lipschitz robustness problem when function and distance are both realized by an $\mathbb{N}$-WA.

Definition : Automata convolution

Let $\mathcal{A} := \langle \Sigma, Q_\mathcal{A}, s_\mathcal{A}, F_\mathcal{A}, \Delta_\mathcal{A}, \gamma_\mathcal{A} \rangle$ and $\mathcal{B} := \langle \Gamma, Q_\mathcal{B}, s_\mathcal{B}, F_\mathcal{B}, \Delta_\mathcal{B}, \gamma_\mathcal{B} \rangle$ two $\mathbb{Z}$Sum-WA, we define $\mathcal{A} \otimes \mathcal{B} := \langle \Sigma \otimes \Gamma, Q_{\mathcal{A}\otimes\mathcal{B}}, s_{\mathcal{A}\otimes\mathcal{B}}, F_{\mathcal{A}\otimes\mathcal{B}}, \Delta_{\mathcal{A}\otimes\mathcal{B}}, \gamma_{\mathcal{A}\otimes\mathcal{B}} \rangle$ such that $\mathcal{L}_{[\mathcal{A}\otimes\mathcal{B}]} = (\mathcal{L}_\mathcal{A} \cap \mathcal{L}_\mathcal{B})^2 \wedge \forall u \otimes v \in \mathcal{L}_{[\mathcal{A}\otimes\mathcal{B}]}.\ [\![\mathcal{A} \otimes \mathcal{B}]\!](u \otimes v) = [\![\mathcal{A}]\!](u) + [\![\mathcal{B}]\!](v)$ and where :

- $Q_{\mathcal{A}\otimes\mathcal{B}} := (Q_\mathcal{A} \cup \{\top\ ;\ \bot\}) \times (Q_\mathcal{B} \cup \{\top\ ;\ \bot\})$
- $s_{\mathcal{A}\otimes\mathcal{B}} := (s_\mathcal{A}, s_\mathcal{B})$
- $F_{\mathcal{A}\otimes\mathcal{B}} := (F_\mathcal{A} \times F_\mathcal{B}) \cup (\{\top\} \times F_\mathcal{B}) \cup (F_\mathcal{A} \times \{\top\})$
- $\Delta_{\mathcal{A}\otimes\mathcal{B}}$ and $\gamma_{\mathcal{A}\otimes\mathcal{B}}$ such that :

1. $p_1 \xrightarrow{a \mid v_1}_\mathcal{A} q_1 \Rightarrow (p_1, \top) \xrightarrow{\langle a,\# \rangle \mid v_1}_{\mathcal{A}\otimes\mathcal{B}} (q_1, \top)$
2. $p_1 \xrightarrow{a \mid v_1}_\mathcal{A} q_1 \Rightarrow (p_1, \bot) \xrightarrow{\langle a,\# \rangle \mid v_1}_{\mathcal{A}\otimes\mathcal{B}} (q_1, \bot)$
4. $p_2 \xrightarrow{b \mid v_2}_\mathcal{B} q_2 \Rightarrow (\top, p_2) \xrightarrow{\langle \#,b \rangle \mid v_2}_{\mathcal{A}\otimes\mathcal{B}} (\top, q_2)$
5. $p_2 \xrightarrow{b \mid v_2}_\mathcal{B} q_2 \Rightarrow (\bot, p_2) \xrightarrow{\langle \#,b \rangle \mid v_2}_{\mathcal{A}\otimes\mathcal{B}} (\bot, q_2)$
6. $p_1 \xrightarrow{a \mid v_1}_\mathcal{A} q_1 \wedge p_2 \xrightarrow{b \mid v_2}_\mathcal{B} q_2 \Rightarrow (p_1, p_2) \xrightarrow{\langle a,b \rangle \mid v_1+v_2}_{\mathcal{A}\otimes\mathcal{B}} (q_1, q_2)$
7. $p_1 \xrightarrow{a \mid v_1}_\mathcal{A} q_1 \wedge p_2 \in F_\mathcal{B} \Rightarrow (p_1, p_2) \xrightarrow{\langle a,\# \rangle \mid v_1}_{\mathcal{A}\otimes\mathcal{B}} (q_1, \top)$
8. $p_1 \xrightarrow{a \mid v_1}_\mathcal{A} q_1 \wedge p_2 \notin F_\mathcal{B} \Rightarrow (p_1, p_2) \xrightarrow{\langle a,\# \rangle \mid v_1}_{\mathcal{A}\otimes\mathcal{B}} (q_1, \bot)$
9. $p_1 \in F_\mathcal{A} \wedge p_2 \xrightarrow{b \mid v_2}_\mathcal{B} q_2 \Rightarrow (p_1, p_2) \xrightarrow{\langle \#,b \rangle \mid v_2}_{\mathcal{A}\otimes\mathcal{B}} (\top, q_2)$
10. $p_1 \notin F_\mathcal{A} \wedge p_2 \xrightarrow{b \mid v_2}_\mathcal{B} q_2 \Rightarrow (p_1, p_2) \xrightarrow{\langle \#,b \rangle \mid v_2}_{\mathcal{A}\otimes\mathcal{B}} (\bot, q_2)$

Intuitively, we just make a classical product with four additional states which memorize the acceptance of the smallest word. We can prove the property $[\![\mathcal{A} \otimes \mathcal{B}]\!](u \otimes v) = [\![\mathcal{A}]\!](u) + [\![\mathcal{B}]\!](v)$ by sum commutativity with a straightforward double induction on the size of words. Remark, if $\mathcal{A}$ and $\mathcal{B}$ are functional then $\mathcal{A} \otimes \mathcal{B}$ is also.

Theorem 5.1

Given $K > 0$, it is undecidable to determine whether a function f performed by an $\mathbb{N}$-WA is K-lipschitz robust for a distance realized by an $\mathbb{N}$-WA.

Proof

By reduction from the inclusion problem for $\mathbb{N}$-WA. Let $\mathcal{A}, \mathcal{B}$ be two total $\mathbb{N}$-WA, $\forall w \in \Sigma^*.\ [\![\mathcal{A}]\!](w) \leq [\![\mathcal{B}]\!](w)$ is an undecidable problem [1]. We build $\mathcal{A}', \mathcal{B}'$ two total $\mathbb{N}$-WA such that $\forall u, v \in \Sigma^*.\ |[\![\mathcal{A}]\!]'(u) - [\![\mathcal{A}]\!]'(v)| \leq K \cdot [\![\mathcal{B}]\!]'(u \otimes v) \iff \forall w \in \Sigma^*.\ [\![\mathcal{A}]\!](w) \leq [\![\mathcal{B}]\!](w)$ where $\mathcal{B}'$ is a distance.

$$K := 1 \qquad \mathcal{A}' := \mathcal{A}^{+1} \qquad \mathcal{B}' := \begin{cases} \mathcal{Z} & \text{if } u = v \\ [\mathcal{B} \otimes \mathcal{B}]^{+1}_{u \neq v} & \text{otherwise} \end{cases} \qquad \text{where } {}^{+1} \text{ means, add 1 to every weight}$$

1. Automata semantic.
   The effect of adding 1 to every weight is to add the size of the word to the return value. Since $\mathcal{B}'$ reads two words, the size of the convolution is the maximal one. Note that $\mathcal{B}'$ just adds non-determinism between $[\mathcal{B} \otimes \mathcal{B}]^{+1}_{u \neq v}$ and $\mathcal{Z}$, see FIGURE 5.1.

   - $\mathcal{L}_{\mathcal{A}'} = \mathcal{L}_\mathcal{A} = \Sigma^* \wedge \forall u \in \mathcal{L}_\mathcal{A}.\ [\![\mathcal{A}']\!](u) = [\![\mathcal{A}]\!](u) + |u|$
   - $\mathcal{L}_{\mathcal{B}'} = \mathcal{L}_\mathcal{B} = \Sigma^* \wedge \forall u \in \mathcal{L}_\mathcal{B}.\ [\![\mathcal{B}']\!](u) = \begin{cases} 0 & \textbf{if } u = v \\ [\![\mathcal{B}]\!](u) + [\![\mathcal{B}]\!](v) + \max\{|u|, |v|\} & \textbf{otherwise} \end{cases}$

2. Check that $\mathcal{B}'$ is a distance.
   non-negative : $\forall u, v \in \Sigma^*.\ [\![\mathcal{B}]\!]'(u \otimes v) \geq 0$, obvious in $\mathbb{N}$.
   symmetry : $\forall u, v \in \Sigma^*.\ [\![\mathcal{B}]\!]'(u \otimes v) = [\![\mathcal{B}]\!]'(v \otimes u)$, by automata convolution.
   separation : $\forall u, v \in \Sigma^*.\ [\![\mathcal{B}]\!]'(u \otimes v) = 0 \Leftrightarrow u = v$, by construction.
   triangle inequality : Essentially, by automata convolution.

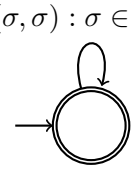


FIGURE 5.1 : $\mathcal{Z}$

$$\forall u, v, w \in \Sigma^*.\ \begin{array}{l} [\![\mathcal{B}]\!]'(u \otimes v) + [\![\mathcal{B}]\!]'(v \otimes w) \geq [\![\mathcal{B}]\!]'(u \otimes w) \iff \\ [\![\mathcal{B}]\!](u) + 2[\![\mathcal{B}]\!](v) + [\![\mathcal{B}]\!](w) + \max\{|u|, |v|\} + \max\{|v|, |w|\} \geq [\![\mathcal{B}]\!](u) + [\![\mathcal{B}]\!](w) + \max\{|u|, |w|\} \end{array}$$

3. Show the equivalence.

   - Assume that $\forall w \in \Sigma^*.\ [\![\mathcal{A}]\!](w) \leq [\![\mathcal{B}]\!](w)$ and let $u, v \in \Sigma^*$. We want to prove that $|[\![\mathcal{A}]\!]'(u) - [\![\mathcal{A}]\!]'(v)| \leq K \cdot [\![\mathcal{B}]\!]'(u \otimes v)$.

$$0 \leq [\![\mathcal{A}]\!](u) \leq [\![\mathcal{B}]\!](u) \Rightarrow 0 \leq [\![\mathcal{A}]\!](u) + |u| \leq [\![\mathcal{B}]\!](u) + [\![\mathcal{B}]\!](v) + \max\{|u| \ ; \ |v|\}$$
$$0 \leq [\![\mathcal{A}]\!](v) \leq [\![\mathcal{B}]\!](v) \Rightarrow 0 \leq [\![\mathcal{A}]\!](v) + |v| \leq [\![\mathcal{B}]\!](u) + [\![\mathcal{B}]\!](v) + \max\{|u| \ ; \ |v|\}$$

Recall that $K := 1$, and then we can deduce, $|[\![\mathcal{A}]\!]'(u) - [\![\mathcal{A}]\!]'(v)| \leq K \cdot [\![\mathcal{B}]\!]'(u \otimes v)$.

- Assume that w satisfies $[\![\mathcal{A}]\!](w) > [\![\mathcal{B}]\!](w)$. We want to prove that there exists u and v such that $|[\![\mathcal{A}]\!]'(u) - [\![\mathcal{A}]\!]'(v)| > K \cdot [\![\mathcal{B}]\!]'(u \otimes v)$. Remark, $[\![\mathcal{A}]\!](\varepsilon) = [\![\mathcal{B}]\!](\varepsilon) = 0$. Thus $w \neq \varepsilon$. We can deduce, $|[\![\mathcal{A}]\!]'(w) - [\![\mathcal{A}]\!]'(\varepsilon)| > K \cdot [\![\mathcal{B}]\!]'(w \otimes \varepsilon)$

We conclude on the undecidability of K-robustness problem by noting that $\mathcal{A}'$ and $\mathcal{B}'$ are total. □

5.2 Decidability of robustness

Now, we define the class $\mathcal{D}$ of functions called "good for distance"and show that functions defined in the previous sections (we say the class $\mathcal{C}$ for short) are in $\mathcal{D}$. Finally, we prove that the lipschitz robustness problem is decidable for a function from $\mathcal{C}$.

Definition : Class $\mathcal{D}$

Let $\mathcal{D}$ the set of pair (f, n) where $f : (\Sigma^*)^n \to \mathbb{R}$ a finitely presented function which satisfies following computable axioms.

1. $\mathcal{D}$ is closed under parameters addition :
   $\forall (f, n) \in \mathcal{D}.\ \exists (g, n+1) \in \mathcal{D}.\ \forall w \in \Sigma^*.\ g(u_1, \ldots, u_n, w) = f(u_1, \ldots, u_n)$
2. $\mathcal{D}$ is closed under parameters permutation :
   $\forall (f, n) \in \mathcal{D}.\ \forall \theta : \{1, \ldots, n\} \to \{1, \ldots, n\}.\ f(\theta(u_1), \ldots, \theta(u_n)) \in \mathcal{D}$
3. $\mathcal{D}$ is closed under sum :
   $\forall (f, n), (g, n) \in \mathcal{D}.\ ([f + g], n) \in \mathcal{D}$ where $[f + g](u_1, \ldots, u_n) := f(u_1, \ldots, u_n) + g(u_1, \ldots, u_n)$
4. Comparison problem is decidable :
   $\forall (f, n), (g, n) \in \mathcal{D}.\ f(u_1, \ldots, u_n) \succ g(u_1, \ldots, u_n)$ where $\succ \in \{=, >, \geq\}$ is decidable.
5. $\mathcal{D}$ is closed under regular domain restriction :
   $\forall (f, n) \in \mathcal{D}.\ \mathcal{L} \subseteq \mathrm{REG}_\Sigma.\ ([f]_\mathcal{L}, n) \in \mathcal{D}$ where $[f]_\mathcal{L}$ the domain restriction of f to $\mathcal{L}$.

Lemma 5.2

Let $(f, 2) \in \mathcal{D}$, one can decide if f is a distance.

Proof

Given $(f, 2) \in \mathcal{D}$, one can check axioms separately.

non-negative : $\forall u, v \in \Sigma^*.\ f(u, v) \geq 0$. Obvious by (4).

symmetry : $\forall u, v \in \Sigma^*.\ f(u, v) = f(v, u)$. Define $g(u, v) := f(v, u)$ by (2) and decide $g(u, v) = f(v, u)$ by (4).

separation : Let $\mathcal{L} := \{(u, v) \in \Sigma^* \times \Sigma^* : u = v\}$. By (5), (4) and with the following equivalence.

$$\forall u, v \in \Sigma^*.\ f(u, v) = 0 \leftrightarrow u = v \iff [f]_\mathcal{L}(u, v) = 0 \wedge [f]_{\overline{\mathcal{L}}}(u, v) > 0$$

triangle inequality : By (4) and with the following equivalence.

$\forall u, v, w \in \Sigma^*.\ f(u, v) + f(v, w) \geq f(u, w)$

$\iff f_1(u, v, w) + f_1(v, w, u) \geq f_1(u, w, v)$ by (1) where $\forall u, v, w \in \Sigma^*.\ f_1(u, v, w) := f(u, v)$

$\iff f_1(u, v, w) + f_2(u, v, w) \geq f_3(u, v, w)$ by (2) where f_2 and f_3 are f_1 with parameters permutation $\theta_2 := \{1 \to 3, 2 \to 1, 3 \to 2\}$ & $\theta_3 := \{1 \to 1, 2 \to 3, 3 \to 2\}$

$\iff [f_1 + f_2](u, v, w) \geq f_3(u, v, w)$ by (3)

□

Definition : Class $\mathcal{C}$

In the previous sections we defined several functions from Σ^* to $\mathbb{Z}$ realized by composition of a valuation and a combiner. All these functions made this new class, see Appendix C. Recall that these functions are closed under min, max, plus and minus operations. Regarding combiner it is obvious for expressions and proved for existential Presburger formulae in lemma 4.5.

Lemma 5.3

The functions of the class $\mathcal{C}$ satisfies the axioms of the class $\mathcal{D}$.

Proof

(1) & (2) are a consequence of word convolution. (3) & (4) have been shown in previous sections. (5) is the construction at the beginning of this section. □

Corollary 5.4

By lemma 5.2 and lemma 5.3, one can decide if a function in $\mathcal{C}$ is a distance (with the complexity of comparison for lower and upper bound).

Theorem 5.5

Given $K > 0$, the K-Lipschitz Robustness for function and distance in $\mathcal{C}$ is decidable (with the complexity of comparison for lower and upper bound).

Proof

This is straightforward, just recall that function of $\mathcal{C}$ class are closed by min, max, plus, minus and rewrite the problem with these operations. In the following equivalence, f_l and f_r add an useless parameter respectively on right and left. Formally, $\forall u, v \in \Sigma^*.\ f_l(u \otimes v) := f(u)$ and $\forall u, v \in \Sigma^*.\ f_r(u \otimes v) := f(v)$.

$$\forall u, v \in dom(f).\ |f(u) - f(v)| \leq K \times d(u, v) \iff$$

$$\forall u, v \in dom(f).\ \max\{f_l(u \otimes v) - f_r(u \otimes v), f_l(u \otimes v) - f_r(u \otimes v)\} \leq \sum_{1}^{K} d(u \otimes v)$$ □

6 Application

We present a simple instance of the problem inspired from [4, 3], and an application in both cases lipschitz robust and not lipschitz robust. Consider a model of a shared resource between two clients A and B. To request access to the resource, clients use the respective input symbols a and b. The system must satisfy the following safety requirements. First, it cannot assign the resource to both clients simultaneously. In LTL syntax, this can be written as $\mathbf{sys}_1 := G\ \neg(A \wedge B)$. Second, a request has to be followed immediately by a grant, which can be formalized by the guarantees $\mathbf{sys}_2 := G\ (a \rightarrow XA)$ and $\mathbf{sys}_3 := G\ (b \rightarrow XB)$. Finally, it is assumed that the environment never raises both request signals at the same time, $\mathbf{env} := G\ \neg(a \wedge b)$. Combining the three guarantees and the assumptions results in the specification $\phi = \mathbf{env} \rightarrow (\mathbf{sys}_1 \wedge \mathbf{sys}_2 \wedge \mathbf{sys}_3)$. It requires the system to satisfy all three guarantees, if the assumption is fulfilled.

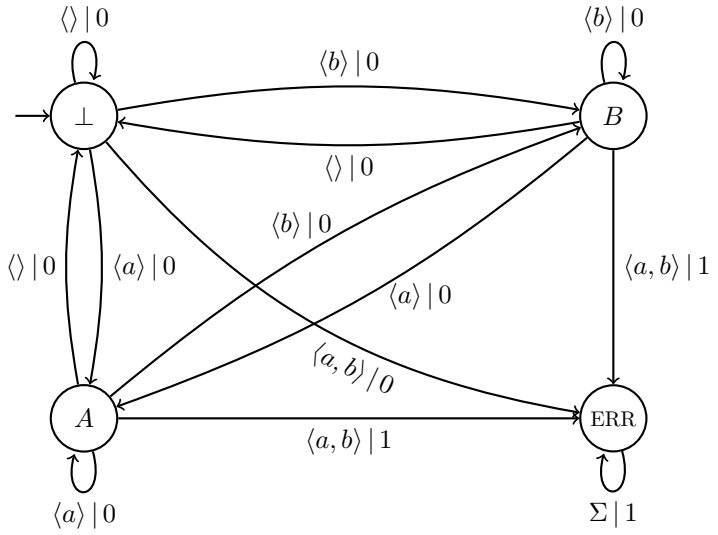


figure 6.1 : $f_{bad} := [\![\mathcal{A}_{bad}]\!]$

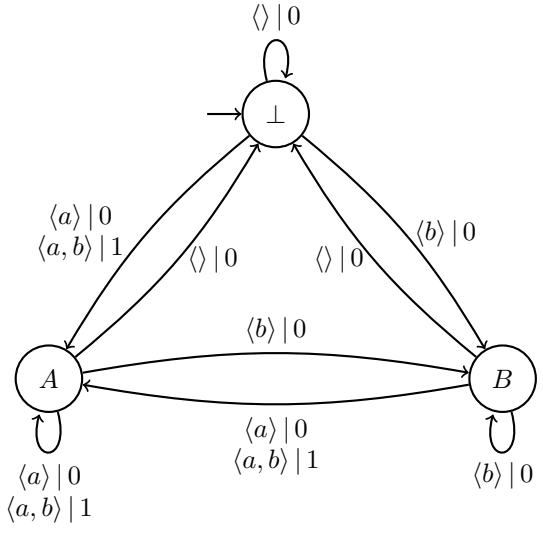


figure 6.2 : $f_{good} := [\![\mathcal{A}_{good}]\!]$

We present $\mathcal{A}_{bad}$, $\mathcal{A}_{good}$, two possible implementations of ϕ in FIGURE 6.1 and FIGURE 6.2 with transitions weighted by 1 if the system makes an error and 0 otherwise. The difference between both is that $\mathcal{A}_{bad}$ crashes when **env** is not satisfied while $\mathcal{A}_{good}$ gives priority to the client A in case of conflict. We will be interested in robustness of these implementations overlooked of environment assumption failure, i.e. the robustness of the implementations behavior when a and b are raised at the same time. Then, we define a distance realized by the expression $[\![d]\!] : u \otimes v \mapsto \max\{[\![\mathcal{A}_l]\!](u \otimes v) - [\![\mathcal{A}_r]\!](u \otimes v), [\![\mathcal{A}_r]\!](u \otimes v) - [\![\mathcal{A}_l]\!](u \otimes v)\}$. Intuitively, $[\![d]\!]$ is the distance between the two input words with the number of times that **sys** is not satisfied metric.

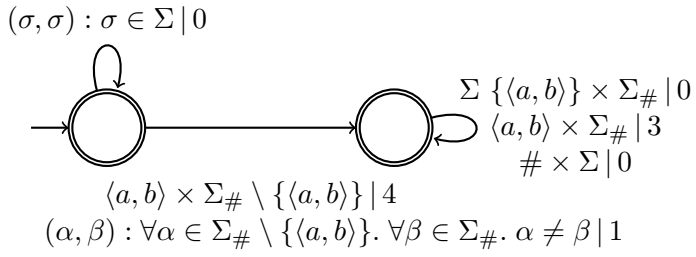


FIGURE 6.3 : $\mathcal{A}_l$

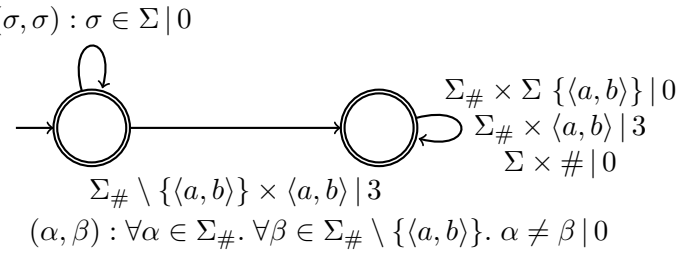


FIGURE 6.4 : $\mathcal{A}_r$

We can check that $[\![d]\!]$ is a distance :

non-negative : $\forall u, v \in \Sigma^*.\ [\![d]\!](u \otimes v) \geq 0$, because $[\![d]\!](u \otimes v) = |[\![\mathcal{A}_l]\!](u \otimes v) - [\![\mathcal{A}_r]\!](u \otimes v)|$.

symmetry : $\forall u, v \in \Sigma^*.\ [\![d]\!](u \otimes v) = [\![d]\!](v \otimes u)$, by commutativity of max function.

separation : $\forall u, v \in \Sigma^*.\ [\![d]\!](u \otimes v) = 0 \Leftrightarrow u = v$, by construction. If $u \neq v$ then $\mathcal{A}_l$ take the middle transition which add 1 to the result and thus $|1 \mod 3 - 0 \mod 3| \neq 0$. If $u = v$ then $\mathcal{A}_l$, $\mathcal{A}_r$ both stay in the first state and return 0.

triangle inequality : $\forall u, v, w \in \Sigma^*.\ [\![d]\!](u \otimes v) + [\![d]\!](v \otimes w) \geq [\![d]\!](u \otimes w)$, because $\forall x, y \in \mathbb{R}.\ |x| + |y| \geq |x + y|$.

Since the value returned by f_{bad} is not linear with respect to the number of environment failures, we can say that f_{bad} is not robust. Formally, set $u \in \langle a, b\rangle\langle a\rangle^*$ such that $|u| = e^{4K}$ we obtain $|f_{bad}(u) - f_{bad}(\varepsilon)| = e^{4K}$ and $K \times [\![d]\!](u \otimes \varepsilon) = 4K$. Then there no $K > 0$ such that $\forall u, v \in \Sigma^*.\ |f_{bad}(u) - f_{bad}(v)| \leq K \times [\![d]\!](u \otimes v)$. On the other hand, f_{good} is 1-Lipschitz robust, in fact we have $\forall u, v \in \Sigma^*.\ 3|f_{good}(u) - f_{good}(v)| - 1 \leq [\![d]\!](u \otimes v) \leq 3|f_{good}(u) - f_{good}(v)| + 1$.

7 Conclusion

In this report, we define a calculation model strictly more expressive than the weighted automata k -valued which all classical quantitative decision problems (emptiness, universality, inclusion, equivalence) are in PSPACE. All complexity results are summarized in the table APPENDIX C. We also presented two general characterizations to obtain the decidability of lipschitz robustness and distance axioms.

A similar analysis could be undertaken if the evaluators are made by discounted sum automata or automata ratio. We could investigate the decidability of games with Presburger winning condition. investigate the decidability of games with Presburger winning condition. On the side of the robustness many works already exist but are not coupled with this model. We might consider the synthesis of robust systems from a logical specification.

We could also consider that our combiners have a expressiveness to heard something closer to a programming language including for example branching (**if**, **then**, **else**). Of course, the behavior of software systems are rarely continuous. However, verification technique like ours allows us to identify modules of a program that satisfy continuity properties. For example the sorting algorithms. Given A, B, two input arrays where B is obtained by perturbing each item of A at most by ± 1. Then, $\mathbf{Sort}(B)$ can be obtained by perturbing each item of $\mathbf{Sort}(A)$ at most by $\pm$ 1.

input	$A : 1, 3, 5, 3, 4$	$B : 1, 2, 4, 4, 3$
output	$\mathbf{Sort}(A) : 1, 3, 3, 4, 5$	$\mathbf{Sort}(B) : 1, 2, 3, 4, 4$

A benefit of this is that such modules are amenable to analysis by continuous methods. In the longer run, we can imagine a reasoning toolkit for programs that combines continuous analysis techniques, like numerical optimization or symbolic integration, and logical methods for analyzing code.

Appendix A : References

[1] Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? In *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings*, pages 482–491, 2011.

[2] Jean Berstel and Christophe Reutenauer. Recognizable formal power series on trees. *Theor. Comput. Sci.*, 18:115–148, 1982.

[3] Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, Georg Hofferek, Barbara Jobstmann, Bettina Könighofer, and Robert Könighofer. Synthesizing robust systems. *Acta Inf.*, 51(3-4):193–220, 2014.

[4] Roderick Bloem, Hans-Jürgen Gamauf, Georg Hofferek, Bettina Könighofer, and Robert Könighofer. Synthesizing robust systems with RATSY. In *Proceedings First Workshop on Synthesis, SYNT 2012, Berkeley, California, USA, 7th and 8th July 2012.*, pages 47–53, 2012.

[5] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust weighted timed automata and games. In *Formal Modeling and Analysis of Timed Systems - 11th International Conference, FORMATS 2013, Buenos Aires, Argentina, August 29-31, 2013. Proceedings*, pages 31–46, 2013.

[6] Krishnendu Chatterjee, Laurent Doyen, Herbert Edelsbrunner, Thomas A. Henzinger, and Philippe Rannou. Mean-payoff automaton expressions. In *CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings*, pages 269–283, 2010.

[7] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. In *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, pages 385–400, 2008.

[8] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Alternating weighted automata. In *Fundamentals of Computation Theory, 17th International Symposium, FCT 2009, Wroclaw, Poland, September 2-4, 2009. Proceedings*, pages 3–13, 2009.

[9] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Algorithms for omega-regular games with imperfect information. In *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, pages 287–302, 2006.

[10] Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublinerman. Continuity and robustness of programs. *Commun. ACM*, 55(8):107–115, 2012.

[11] Thomas Colcombet and Laure Daviaud. Approximate comparison of functions computed by distance automata. *Theory Comput. Syst.*, 58(4):579–613, 2016.

[12] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 513–525, 2005.

[13] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[14] Samuel Eilenberg. *Automata, languages, and machines. Vol. B.* Academic Press [Harcourt Brace Jovanovich Publishers], New York, 1976. With two chapters ("Depth decomposition theorem" and "Complexity of semigroups and morphisms") by Bret Tilson.

[15] Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Quantitative languages defined by functional automata. *Logical Methods in Computer Science*, 11(3), 2015.

[16] M. R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4(4):397–411, 1975.

[17] Eitan M. Gurari and Oscar H. Ibarra. The complexity of decision problems for finite-turn multicounter machines. In *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings*, pages 495–505, 1981.

[18] Eitan M. Gurari and Oscar H. Ibarra. The complexity of the equivalence problem for two characterizations of presburger sets. *Theor. Comput. Sci.*, 13:295–314, 1981.

[19] Arie Gurfinkel and Marsha Chechik. Multi-valued model checking via classical model checking. In *CONCUR 2003 - Concurrency Theory, 14th International Conference, Marseille, France, September 3-5, 2003, Proceedings*, pages 263–277, 2003.

[20] Thomas A. Henzinger, Jan Otop, and Roopsha Samanta. Lipschitz robustness of finite-state transducers. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 431–443, 2014.

[21] Thomas A. Henzinger, Jan Otop, and Roopsha Samanta. Lipschitz robustness of timed I/O systems. In *Verification, Model Checking, and Abstract Interpretation - 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, January 17-19, 2016. Proceedings*, pages 250–267, 2016.

[22] Neil D. Jones. Space-bounded reducibility among combinatorial problems. *J. Comput. Syst. Sci.*, 11(1):68–85, 1975.

[23] R. Karp N. Karmarkar. The differencing method of set partitioning. Technical Report UCB/CSD 82/113, University of California, Berkeley, 1982.

[24] Stephen Cole Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, 1956.

[25] Dexter Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 254–266, 1977.

[26] Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *IJAC*, 4(3):405–426, 1994.

[27] Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory, College Park, Maryland, USA, October 25-27, 1972*, pages 125–129, 1972.

[28] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[29] Michael Oser Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[30] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages, Vol. 1: Word, Language, Grammar.* Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[31] Jacques Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, 2009.

[32] Roopsha Samanta, Jyotirmoy V. Deshmukh, and Swarat Chaudhuri. Robustness analysis of string transducers. In *Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, pages 427–441, 2013.

[33] Bruno Scarpellini. Complexity of subcases of presburger arithmetic. In *Transactions of the American Mathematical Society 284*, page 203–218, 1984.

[34] Marcel Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.

[35] Ken Thompson. Regular expression search algorithm. *Commun. ACM*, 11(6):419–422, 1968.

[36] Yaron Velner. The complexity of mean-payoff automaton expression. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 390–402, 2012.

[37] Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the complexity of equational horn clauses. In *Automated Deduction - CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings*, pages 337–352, 2005.

[38] Yasushige Watase. Lagrange's four-square theorem. *Formalized Mathematics*, 22(2):105–110, 2014.

[39] Andreas Weber. *A decomposition theorem for finite-valued transducers and an application to the equivalence problem*, pages 552–562. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.

Appendix B : Details

LEMMA 3.1 : We wants create a macro to do $p \xrightarrow{\sigma|x_1+c_1;\dots;x_k+c_k}_{\mathcal{M}} q$ with $k > 0$ and $x_j := x_j^+$ or $x_j := x_j^-$ according to sign of c_j for each j. The transition is illegal because a non-constant number counters can be transformed simultaneously. To do this, we split it in more simpler macros. Note that the following constructions do not decrease counter and all zero guards are **top**.

- We describe a first macro for $p_{(i,0)} \xrightarrow{\varepsilon|x_i+c}_{\mathcal{M}} p_{(i,|c|)}$. According to sign of the constant c :

  $\boxed{c=0}$ $p_{(i,0)}$

  $\boxed{c>0}$ $p_{(i,0)} \xrightarrow{\varepsilon|x_i^++1}_{\mathcal{M}} p_{(i,1)} \cdots p_{(i,c-1)} \xrightarrow{\varepsilon|x_i^++1}_{\mathcal{M}} p_{(i,c)}$

  $\boxed{c<0}$ $p_{(i,0)} \xrightarrow{\varepsilon|x_i^-+1}_{\mathcal{M}} p_{(i,1)} \cdots p_{(i,|c|-1)} \xrightarrow{\varepsilon|x_i^-+1}_{\mathcal{M}} p_{(i,|c|)}$

  Note that this construction has $c+1$ states and c transitions

- We describe a second macro for $p_{(1,0)} \xrightarrow{\varepsilon|x_1+c_1;\dots;x_k+c_k}_{\mathcal{M}} p_{(k,|c_k|)}$. By induction on $k > 0$:

  $\boxed{k=1}$ $p_{(1,0)} \xrightarrow{\varepsilon|x_1+c_1}_{\mathcal{M}} p_{(1,|c_1|)}$

  $\boxed{k>1}$ $p_{(1,0)} \xrightarrow{\varepsilon|x_1+c_1;\dots;x_{k-1}+c_{k-1}}_{\mathcal{M}} p_{(k-1,|c_{k-1}|)} \xrightarrow{\varepsilon} p_{(k,0)} \xrightarrow{\varepsilon|x_k+c_k}_{\mathcal{M}} p_{(k,|c_k|)}$

  Note that this construction has $k + \sum_{j=1}^{k} c_j$ states and $(k-1) + \sum_{j=1}^{k} c_j$ transitions.

- We describe a third macro for $p \xrightarrow{\sigma|x_1+c_1;\dots;x_k+c_k}_{\mathcal{M}} q$.
  $p \xrightarrow{\sigma} p_{(1,1)} \xrightarrow{\varepsilon|x_1+c_1;\dots;x_k+c_k}_{\mathcal{M}} p_{(k,|c_k|)} \xrightarrow{\varepsilon} q$

  Note that this construction has $2 + k + \sum_{j=1}^{k} c_j$ states and $k + 1 + \sum_{j=1}^{k} c_j$ transitions.

PRESBURGER ARITHMETIC : To be convinced that Presburger arithmetic with variables interpreted over natural or rationnal number have same expressiveness, we give an encoding of a domain in the other model and a formula describing all the correct encodings.

$\boxed{\mathbb{N} \to \mathbb{Z}}$ We propose an encoding (not injective because there are two 0) which consists in representing a rational by a natural couple $x_{\mathbb{Z}} := (x_{\text{sign}}, x_{\text{val}})$. Other relations can be defined as follow.

$$x_{\mathbb{Z}} =_{\mathbb{Z}} y_{\mathbb{Z}} := (x_{\text{val}} = y_{\text{val}} = 0) \vee (x_{\text{sign}} = y_{\text{sign}} \wedge x_{\text{val}} = y_{\text{val}})$$

$$x_{\mathbb{Z}} <_{\mathbb{Z}} y_{\mathbb{Z}} := (x_{\text{sign}} = y_{\text{sign}}) \vee (x_{\text{sign}} = y_{\text{sign}} = 1 \wedge x_{\text{val}} < y_{\text{val}}) \vee (x_{\text{sign}} = y_{\text{sign}} = 0 \wedge y_{\text{val}} < x_{\text{val}})$$

$$z_{\mathbb{Z}} =_{\mathbb{Z}} x_{\mathbb{Z}} +_{\mathbb{Z}} y_{\mathbb{Z}} := \bigwedge \begin{pmatrix} x_{\text{val}} < y_{\text{val}} \vee y_{\text{val}} < x_{\text{val}} \vee z_{\text{sign}} = x_{\text{sign}} \\ x_{\text{val}} < y_{\text{val}} \vee z_{\text{sign}} = x_{\text{sign}} \\ y_{\text{val}} < x_{\text{val}} \vee z_{\text{sign}} = y_{\text{sign}} \\ x_{\text{sign}} < y_{\text{sign}} \vee y_{\text{sign}} < x_{\text{sign}} \vee z_{\text{val}} = x_{\text{val}} + y_{\text{val}} \\ x_{\text{sign}} < y_{\text{sign}} \vee z_{\text{val}} + y_{\text{val}} = x_{\text{val}} \\ y_{\text{sign}} < x_{\text{sign}} \vee z_{\text{val}} + x_{\text{val}} = y_{\text{val}} \end{pmatrix}$$

$\boxed{\mathbb{Z} \to \mathbb{N}}$ Since $\mathbb{N} \subseteq \mathbb{Z}$ there is no need encoding, so $x_{\mathbb{N}} := x$. It is possible to describe the set of positive numbers with four squares theorem of Lagrange [38] $\exists a, b, c, d.\ x_{\mathbb{N}} = a + a + b + b + c + c + d + d$. The other relations are completely obvious.

Appendix C : Summary

<table>
<tr><th>$\Sigma^* \to \mathbb{Z}^k$</th><th>$\mathbb{Z}^k \to \mathbb{Z}$</th><th>fixed parameters</th><th>encoding</th><th>$\exists$</th><th>$\forall$</th><th>$\succ$</th></tr>
<tr><td rowspan="6">functional
$\mathbb{Z}^k$Sum-WA</td><td rowspan="4">Expr</td><td rowspan="2"></td><td>binary</td><td>NP – Complete</td><td>CoNP – Complete</td><td></td></tr>
<tr><td>unary</td><td>NP – Complete</td><td>CoNP – Complete</td><td></td></tr>
<tr><td rowspan="2">$k, |E|$</td><td>binary</td><td>NP – Complete</td><td>CoNP – Complete</td><td></td></tr>
<tr><td>unary</td><td colspan="3">NLogSpace – Complete</td></tr>
<tr><td rowspan="2">Presburger</td><td rowspan="2"></td><td>binary</td><td rowspan="2">NP−Complete</td><td rowspan="2">CoNP−Complete</td><td></td></tr>
<tr><td>unary</td><td></td></tr>
<tr><td rowspan="6">functional
$\prod^k \mathbb{Z}$Sum-WA</td><td rowspan="4">Expr</td><td rowspan="2"></td><td>binary</td><td colspan="3">PSpace – Complete</td></tr>
<tr><td>unary</td><td colspan="3"></td></tr>
<tr><td rowspan="2">$k, |E|$</td><td>binary</td><td>NP – Complete</td><td>CoNP – Complete</td><td></td></tr>
<tr><td>unary</td><td colspan="3">NLogSpace – Complete</td></tr>
<tr><td rowspan="2">Presburger</td><td rowspan="2"></td><td>binary</td><td colspan="3" rowspan="2">PSpace – Complete</td></tr>
<tr><td>unary</td></tr>
</table>